IMPROVING MACHINE LEARNING APPROACHES TO NOUN PHRASE COREFERENCE RESOLUTION

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Yu-Chung Ng

August 2004

IMPROVING MACHINE LEARNING APPROACHES TO NOUN PHRASE COREFERENCE RESOLUTION

Yu-Chung Ng, Ph.D.

Cornell University 2004

Human speakers generally have no difficulty in determining which noun phrases in a text or dialogue refer to the same real-world entity. This task of identifying co-referring noun phrases — noun phrase coreference resolution — can present a serious challenge to a natural language processing system, however. Indeed, it is one of the critical problems that currently limits the performance for many practical natural language processing tasks.

State-of-the-art coreference resolution systems operate by relying on a set of hand-crafted heuristics that requires a lot of time and linguistic expertise to develop. Recently, machine learning techniques have been used to circumvent both of these problems by automating the acquisition of coreference resolution heuristics, yielding coreference systems that offer performance comparable to their heuristic-based counterparts. In this dissertation, we present a machine learning-based solution to noun phrase coreference that extends earlier work in the area and outperforms the best existing learning-based coreference engine on a suite of standard coreference data sets. Performance gains accrue from more effective use of the available training data via a set of linguistic and extra-linguistic extensions to the standard machine learning framework for coreference resolution.

BIOGRAPHICAL SKETCH

Yu-Chung Ng was born and raised in Hong Kong, a well-known former British colony in southern China. He had a very happy childhood, during which his parents fostered his extra-curricular interests by bringing him to drawing and piano lessons.

Computers only played a minor role during his teenage years. Back then home computers were not in widespread use, and although his parents were generous enough to get him an Apple II-E computer and later an Intel 486, he used them primarily for playing computer games. Moreover, although he studied many science subjects in primary and secondary schools, he never had a formal course in computing or information technology.

In August 1994 he came to the United States to begin his study of computer science at Carnegie Mellon University as an undergraduate. There he wrote his first computer program and developed a sincere interest in this rapidly growing discipline. After earning his B.S. in computer science in 1997, he moved to Ithaca to continue his quest for knowledge at Cornell University. A year later he obtained a professional Masters degree from the computer science department. During his Masters study, he met Keshav Pingali, one of his favorite professors at Cornell. Upon Keshav's enthusiastic recommendation, he was admitted into the department's doctoral program and spent the following six years studying natural language processing.

In September he will start work as an assistant professor at the University of Texas at Dallas. He looks forward eagerly to getting involved in building a first-class natural language processing group at Dallas. Being far away from Hong Kong, he dearly misses the city's good food and great weather, as well as his friends and family there.

ACKNOWLEDGMENTS

Like many other people, acknowledgments is the last section I work on in the writing of the dissertation. It reminds me that my Cornell days are coming close to an end. Perhaps this is the time for me to acknowledge the people who have helped me reach where I am now.

First of all, I would like to thank my advisor, Claire Cardie, for allowing a student who did not really know how to write code properly to work on her large-scale NLP projects. Though I never learned to become a good programmer, I inherited from Claire a passion for research in natural language learning. I want to take this opportunity to thank her for what she has done for me.

Lillian Lee introduced me to the realm of statistical natural language processing. She was very supportive during these years, attending virtually all of my talks and giving invaluable feedback. In Lillian I see a researcher who is very confident of her work. I have a lot of respect for her.

Mats Rooth exposed me to the wonders of statistical parsing research. I have always hoped to possess his profound knowledge in linguistics and mathematics. I would like to thank him for numerous interesting and inspiring discussions.

Molly Diesing kindly agreed to serve as my minor committee member even before I started to take courses for my linguistics minor. She was too nice to say that "this is the responsibility of a professor". I am also grateful to Sally McConnell-Ginet for agreeing to act as a proxy for Molly during my defense.

I have a special feeling towards the Cornell NLP group. It grows and matures with me. I am glad to be able to witness several historic events of the group: the graduation of our first-generation NLP students, Claire and Lillian's becoming permanent fixtures of the department, as well as the expansion of the group with

truly brilliant students who I believe can help strengthen its reputation. I take great pride in being its member.

I have had the opportunity to work on various fun projects with former group members David Pierce and Kiri Wagstaff. Dave taught me all I needed to know about Lisp. From Kiri I learned how to get things done efficiently.

One of my favorite pastimes was to chat with our new-generation NLP students including Eric Breck, Oren Kurland, Bo Pang, and Ves Stoyanov. I want to thank them for updating me on the news about the group in a timely manner.

I have also had some memorable times with my officemates. My thanks go to Yu Zhang and Li Li (for spending countless nights with me working in the office), Rie Kubota Ando (for amusing chats outside Rhodes on the chilly days), and Rohit Fernandes (for his willingness to listen to me whenever I need someone to talk to).

Several people have helped make my life at Cornell more enjoyable: Ken Hopkinson, Yonghong Mao, Hui Tan, Aggrey Wasike, Wei Wei, Wes Weimer, Changguk Yim, and Lantian Zheng.

Regina Barzilay deserves special mention. She became one of my closest buddies at Cornell, both professionally and non-professionally, soon after we got to know each other. She convinced me that there are lots of interesting problems to work on in NLP. I truly admire her enthusiasm for research, and appreciate her willingness to share her extremely creative ideas with me.

Though far away from me, my dear old friends have cheered me up via their frequent e-mails and unexpected phone calls. I am thankful to Arnold Leung, Jacky Low, Jacky Poon, Simon Chan, Matt Ho, Sunny Im, and Lawrence Shiou.

Above all, my sincerest gratitude to my mom and dad, and my brother Alan and his wife Carol. They were always there to help me get through my stressful

times. This dissertation is dedicated to them.

This work was supported in part by DARPA TIDES contract N66001-00-C-8009 and NSF grants IIS-0081334, EIA-0074896, and IIS-0208028. Any opinions, findings, and conclusions or recommendations expressed below are those of the author and do not necessarily reflect the views of the Defense Advanced Research Projects Agency or the National Science Foundation.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Natural language applications have always been an integral part of research in natural language processing (NLP). In fact, the classic natural language application — machine translation — has roots dating back to the 1940s, even before the terms "computational linguistics" and "natural language processing" existed. Nowadays, a number of natural language applications are in constant use by millions of online users every day. For example, programs for machine translation of one natural language to another are commonly used to translate e-mails and web pages. More recently, question answering systems such as BrainBoost[1] have been built to locate specific pieces of information from a large text database in response to users' queries. Forming the backbone of these natural language applications is a set of components for performing text analysis at various linguistic levels. The kind of text analysis may range from morphological analysis — which is the study of how words are constructed from basic meaning units — to discourse analysis — which concerns how the interpretation of a sentence is affected by its immediately preceding sentences.

In this dissertation, we examine in detail a key task in discourse analysis and in many natural language applications — noun phrase coreference resolution. Informally, the goal of coreference resolution is to determine which noun phrases refer to each real-world entity mentioned in a document. As we will see later in this chapter, coreference resolution is generally considered one of the most difficult problems in NLP, and many important natural language applications can potentially benefit from the availability of coreference information. In fact, a number

[1] `www.brainboost.com`

of coreference resolution systems have been developed during the past few years. Unfortunately, virtually none of these systems has attained usable accuracy levels for state-of-the-art natural language applications. *The goal of this dissertation is to improve existing approaches to coreference resolution.* Specifically, we focus on improving a class of approaches that has been applied reasonably successfully to the problem — *machine learning approaches*.

The rest of this chapter is organized as follows. In Section 1.1, we define coreference resolution via an example, and compare coreference with a closely related linguistic phenomenon known as anaphora. Sections 1.2 and 1.3 provide an empirical justification of the hardness of coreference resolution and discuss the potential applications of the problem, respectively. In Section 1.4, we discuss two major classes of approaches to coreference resolution — knowledge-based approaches and machine learning approaches — as well as our decision to focus on machine learning approaches. Section 1.5 describes the goals and contributions of this dissertation, distinguishing our research from existing work on coreference resolution. Finally, we provide the roadmap of the dissertation in Section 1.6.

1.1 Noun Phrase Coreference

As mentioned previously, noun phrase coreference resolution refers to the problem of determining which noun phrases refer to each real-world entity mentioned in a document. Throughout the dissertation, we use the term *noun phrases*, or NPs, to refer to three types of phrases: pronouns, definite noun phrases, and common noun phrases, following the convention adopted by the Penn Treebank (Marcus et al., 1993).

Figure 1.1 depicts the input/output behavior of a coreference resolution system.

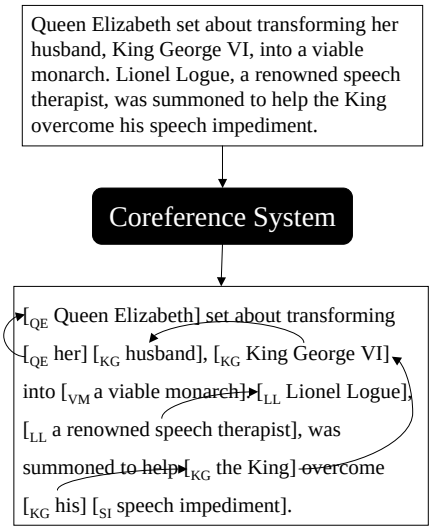

Figure 1.1: Coreference Resolution System

Given an arbitrary text as input, a coreference resolution system partitions its noun phrases in such a way that all noun phrases that refer to the same real-world entity are in the same cluster. In this example, a coreference system should partition the noun phrases in the paragraph into five equivalence classes, generating three coreference chains — QE (*Queen Elizabeth* and *her*), KG (*husband*, *King George VI*, *the King*, and *his*) and LL (*Lionel Logue* and *a renowned speech therapist*) — and two NP singletons, VM (*a viable monarch*) and SI (*speech impediment*). Hence, coreference is an equivalence relation defined on the set of noun phrases.[2] While human audiences have few problems of identifying co-referring NPs, coreference resolution can present a challenge to NLP systems.

1.1.1 Comparison to Anaphora

Coreference resolution bears close resemblance to a classic NLP problem known as anaphora resolution. Before introducing anaphora resolution, let us define some

[2]In other words, the coreference relation is symmetric, reflexive, and transitive.

terminology that we will be using extensively in the rest of this dissertation.

A noun phrase can be a **referring expression**, which depends on other NPs for its semantic interpretation, or a **non-referring expression**, which "selects" or uniquely identifies a referent from a set of a real-world entities that we know and talk about (see Haegeman (1994)). For instance, among the four noun phrases *husband*, *King George VI*, *the King*, and *his* that are used to refer to the discourse referent *King George VI* in our example, only the NP *King George VI* uniquely identifies the referent and is by definition non-referring. On the other hand, each of the remaining noun phrases depends on *King George VI* for its interpretation and therefore is a referring expression. For example, the NP *the husband* only identifies a subset of the real-world entities that satisfy the property of being a husband but does not select a particular entity from the real world.

Now, given an NP pair $<\text{NP}_i, \text{NP}_j>$ such that (1) NP_j is a referring expression, (2) NP_i precedes NP_j in the associated text, and (3) NP_i is coreferent with NP_j, we say that NP_j is an **anaphor** and NP_i is its **antecedent**. The goal of anaphora resolution is to identify an antecedent for each anaphor, or anaphoric noun phrase, in a text.

Although anaphora and coreference are similar in various respects, there are a number of subtle differences between the two linguistic phenomena.[3]

- Anaphora refers to the linguistic phenomenon of having a noun phrase refer to a previously mentioned entity in a text for its semantic interpretation and is therefore a *context-sensitive* relation. In other words, a pair of NPs $<\text{NP}_i, \text{NP}_j>$ constitutes an anaphoric relationship if $i < j$ and NP_j depends on NP_i for

[3]See Hirst (1981) and van Deemter and Kibble (2000) for a more detailed description of the differences between anaphora and coreference.

its interpretation, where NP_k denotes the kth NP in a document. For instance, the NP pair <*Queen Elizabeth*, *her*> forms an anaphoric relationship in our example. On the other hand, coreference refers to the problem of identifying all noun phrases that refer to the same entity in a text and is not necessarily context-sensitive. For example, the noun phrases *Queen Elizabeth* and *the Queen Mother* can both be understood to refer to the same real-world entity in the absence of any contextual information and by definition they enjoy a coreference (but not anaphoric) relationship.

- Anaphora, unlike coreference, is not an equivalence relation. Since an anaphoric entity cannot depend on itself for its semantic interpretation, the anaphoric relation is not reflexive. In addition, the relation is by definition an order-dependent relation and is therefore asymmetric with respect to the two NPs involved. As we have seen before, the pair <*Queen Elizabeth*, *her*> constitutes an anaphoric relationship, but <*her*, *Queen Elizabeth*> does not.[4] Transitivity is undefined for the anaphoric relation, since an NP either refers independently or depends on other NPs for its interpretation.

- Some anaphoric relations are not coreferential. In bound anaphora, the anaphor is not coreferential with the antecedent, as exemplified in the sentence *Each of Diana's former police bodyguards had his opinion about the late princess' private life*, where the pronoun *his* lies within the scope of the universal quantifier *each* and refers to each of the persons that is being quantified over. In other words, *his* is not considered coreferent with *each of*

[4]As a side remark, forward references in which an NP depends on a referential expression that appears after it in the associated text for its interpretation are known as *cataphoric* NPs and are not considered anaphoric.

Diana's former police bodyguards.

- Some coreferential relations are not anaphoric. Anaphora is a within-document phenomenon, whereas cross-document coreference is possible. For example, two occurrences of *Queen Elizabeth* in different documents can be considered coreferent, whereas a pronoun *her* in document x cannot form an anaphoric relationship with *Queen Elizabeth* in document y even if the pronoun refers to *Queen Elizabeth* in document x.

For the most part, however, we will not attempt to make a distinction between anaphora and coreference in this dissertation. In fact, we will restrict our attention to within-document entity coreference, viewing coreference resolution essentially as anaphora resolution, in which we attempt to identify an antecedent for each "anaphoric" noun phrase by searching for a preceding noun phrase for each discourse entity that is potentially coreferent with it. This implies that *any* noun phrase that is part of a coreference chain but is not the head of the chain will be considered "anaphoric" under our definition. Linguistically speaking, however, this definition is by no means a self-imposed limitation on the scope of coreference resolution, as the goal of partitioning a set of discourse entities into coreference equivalence classes remains the same. As we will see, treating all types of noun phrases as potentially anaphoric enables us to develop a conceptually simple solution to the problem and is therefore an advantage from a computational perspective.

1.2 Coreference Resolution as a Difficult Problem

Charniak (1972) demonstrated rather convincingly that in order to do pronoun resolution, one had to be able to do everything else. — Hobbs

(1978)

> *It [Pronoun resolution] is also widely considered to be inherently an 'A.I. complete' task – meaning that resolution of pronouns requires full world knowledge and inference.* — Baldwin (1997)

> *NLP in general is very difficult but after working hard on anaphora resolution we have learned it is particularly difficult.* — Mitkov (2001)

Coreference resolution is an interesting task by itself, partly owing to the inherent difficulty of the problem. While anaphora resolution has drawn considerable interest in the computational linguistics community for more than three decades, the problem is far from being solved. In fact, as reflected by the quotes above, anaphora/coreference resolution has widely been considered to be one of the most challenging problems in artificial intelligence. There has been prevailing consensus that the difficulty of the problem lies in its dependence on sophisticated semantic and world knowledge. Here we will elaborate further on this by providing some justifications with respect to coreference resolution, but the arguments carry over to anaphora resolution as well.

- **Many sources of information play a role.** For instance, lexical information such as head noun matches (as in *Lord Spencer* and *Spencer*) is an indicator of coreference, although it is not an absolute indicator (e.g., *Lord Spencer* and *Diana Spencer* are not coreferent). In addition, syntactic constraints such as the binding constraints place restrictions on the interpretation of different kinds of noun phrases. Furthermore, knowledge sources such as gender and number, semantic class, discourse focus, and world knowledge also play a role in determining whether two discourse entities are coreferent.

- **No single source of knowledge is a completely reliable indicator.** For example, two semantically compatible NPs are potentially coreferent (e.g., *Diana Spencer* and *the princess*), but whether the NPs are actually coreferent depends on other factors (such as contextual information). Additional complications arise from the fact that even traditional linguistic constraints indicating (non-)coreference such as number (dis)agreement are not absolutely hard (e.g., the singular NP *assassination (of her bodyguards)* can be coreferent with the plural NP *these murders*).

- **Coreference strategies differ depending on the type of NP.** Statistically speaking, definite NPs are more likely to be anaphoric than their non-definite counterparts (e.g., the article immediately preceding *reporter* in the sentence *Diana saw the/a reporter following her secretly* determines whether the NP has an existential or definite reading). Furthermore, pronoun resolution, which can be viewed as coreference resolution applicable only to pronouns, is notoriously difficult, partly because resolution strategies differ for each type of pronouns (e.g., reflexives versus possessives) and partly because some pronouns such as pleonastic pronouns are semantically empty and therefore need to be handled separately (e.g., the pronoun *it* in the sentence *Camilla went outside and it was raining* is pleonastic).

If we adopt the view that anaphora resolution is simply coreference resolution applied to a subset of the NPs that the latter problem normally considers, then we can argue that coreference resolution is at least as hard and potentially as interesting as anaphora resolution.

1.3 Coreference Resolution as an Important Problem

Research in coreference resolution in the NLP community is primarily motivated by practical concerns. Indeed, coreference resolution is a key task in many NLP applications; this makes it an important problem in language understanding. For illustrative purposes, we here enumerate several applications can potentially benefit or have actually benefited from the availability of coreference information.

Question answering. In its most general form, the task of question answering is concerned with finding an answer to a natural language question from a large text collection or determining that an answer cannot be found. Question answering systems are useful in that they provide the exact information users need and reduce their burden of having to search through the documents returned by search engines in response to their queries. To see the connection between question answering and coreference resolution, consider the query *Where was Mozart born?*. A question answering system may first retrieve the sentence *He was born in Salzberg* from a document talking about *Mozart*. In this case, the system will return the correct answer if it can determine that the pronoun *He* is coreferent with *Mozart*. In fact, several question answering systems have begun to make use of coreference information and shown improvement in system performance (Breck et al. (1999); Singhal et al. (1999); Morton (2000)). In a question answering system, each sentence in a document that is retrieved for a given question is usually assigned a numeric score based on the sum of the term weight of each overlapping word between the question and the sentence (Singhal et al., 1999), and sentences with the highest scores are assumed to contain potential answers. Morton (2000) augments the above algorithm by considering all the terms that are in the same coreference

chain as any of the noun phrases in a sentence to have appeared in that sentence as well, giving these additional terms 90% of the normal term weight. By altering the ranking of each potential answer based on identified coreference relations, Morton reports improvement in the performance in his coreference system.

Single-document summarization. Summarization helps users focus on or keep track of the "important" information contained in a large pool of documents. Informally, the goal of single-document summarization is to provide a short summary of a document. Because of the usefulness of the task, the newest version of Microsoft Word is capable of single-document summarization, for example. Although most single-document summarization systems have not yet taken into account coreference information, there is initial evidence that it would be helpful to do so. For example, Baldwin and Morton (1998) describe a method for generating extraction-based, query-dependent summaries. Not only do they include in the summary those sentences that contain a term appearing in the query, they also incorporate sentences containing a noun phrase that is coreferent with a term occurring in a sentence already selected by the system.

Generation of referring expressions. Anaphoric expressions are used to improve the coherence of a discourse. For instance, consider the following discourse.

> *Queen Elizabeth II celebrated her golden jubilee in 2002. The Queen ascended the throne in 1952 when her father, King George VI, passed away.*

Note that the referring expressions *the Queen* and both occurrences of *her* are coreferent with the NP *Queen Elizabeth II*. Had we replaced these three referring

expressions with *Queen Elizabeth II*, the discourse would have been perceived as incoherent. Hence, it is crucial for a text generation system to be able to determine when to generate referring expressions. In particular, *pronoun generation* has been an active area of research in the community (see McCoy and Strube (1999)).

Many existing models of pronoun generation operate by assuming the availability of coreference information or knowledge of the referent of a given pronoun. To determine when to pronominalize, traditional text generation systems such as Appelt (1981) and McKeown (1985) adopt the naive heuristic that if the current sentence and the immediately preceding sentence are about the same thing, then use a pronoun in the current sentence to refer to that thing. Dale's (1992) system generates a pronoun if its referent is the discourse focus of the last utterance. More recently, Strube and Wolters (2000) have learned a statistical model of pronominalization, concluding that the distance to the closest referent is an influential factor in determining whether a discourse entity should be pronominalized.

Cross-document entity coreference. The goal of cross-document entity coreference is to determine whether a given pair of noun phrases occurring in different documents refers to the same entity. Cross-document coreference is particularly useful for text summarization systems that need to identify and merge the same piece of information about an entity mentioned in different documents in order to avoid repetition. Bagga and Baldwin (1998) show how within-document coreference information can potentially be used to perform cross-document coreference. Specifically, for any two NPs, NP_i from document 1 and NP_j from document 2, that are potentially cross-document coreferent, the cross-document coreference system first extracts all and only those sentences having a noun phrase that is coreferent

with NP_i in document 1. Another set of sentences is similarly extracted from document 2 using NP_j. The vector space model (Salton et al., 1975) is then applied to compute the similarity between these two "documents". If the similarity score is above a certain threshold, then NP_i and NP_j are considered to be cross-document coreferent.

Information extraction. An information extraction (IE) system takes as input a text relevant to a given domain and automatically extracts specific pieces of information from the text. A number of Internet applications crucially rely on IE technologies, including NLP systems that construct newsgroup query systems (Thompson et al., 1997) and weather forecast databases (Soderland, 1997) from Web pages. The importance of coreference resolution in IE is reflected in the decision to make coreference one of the four evaluated IE sub-tasks in MUC-6 (1995). Specifically, coreference information is used to merge discourse entities in a text, allowing the fusion of extracted information regarding the same entity from different sentences. This information in turn aids *template generation*, which determines the number of events described in the text and maps each piece of information being extracted onto the appropriate event (Cardie, 1997).

Machine translation. There is an obvious need for high-quality machine translation (MT) systems. In most existing MT systems, an anaphor in the source language is simply be translated to the corresponding anaphor in the target language. Anaphora resolution comes into play when discrepancies exist between the two languages with respect to anaphor selection. For example, a pronoun in the Malay language is often translated directly by its antecedent (Mitkov, 1999).

1.4 Why Focus on Machine Learning Approaches?

As mentioned before, our goal is to improve existing machine learning approaches to coreference resolution. In this section, we will explain our decision to focus on machine learning approaches.

Roughly speaking, existing approaches to coreference resolution can be classified as either *knowledge-based* or *learning-based* (a.k.a. corpus-based). Both knowledge-based and learning-based approaches are instantiations of a standard approach to the problem. Below we will first give an overview of this standard approach, followed by a description of knowledge-based and corpus-based approaches as well as the differences between the two. Finally, based on these differences, we will discuss our decision to focus on machine learning approaches.

1.4.1 Coreference Resolution: The Standard Approach

The standard approach to coreference resolution consists of two steps: PREDICTION and CLUSTERING.

The goal of PREDICTION is to classify whether two NPs in a text are coreferent or not. Each such classification decision is made by a prediction procedure, which maps a description of two NPs to one of two labels, COREFERENT or NOT COREFERENT. The description of the two NPs is usually expressed in terms of the knowledge sources potentially useful for correctly predicting its label. For instance, one commonly used knowledge source for predicting coreference is GENDER, which encodes information about whether the two NPs under consideration have the same gender or not. The output of this step, therefore, is a set of pairwise classifications, with one label for each pair of NPs in the text.

Now, recall that the goal of coreference resolution is to partition the NPs in the text into equivalence classes, where each class corresponds to a distinct real-world entity. However, making pairwise coreference classifications does not guarantee that the transitivity constraint inherent in the coreference relation will be observed. For instance, the coreference classifier might determine that A is coreferent with B, and B with C, but that A and C are not coreferent. As a result, a separate CLUSTERING mechanism is used to coordinate the possibly contradictory pairwise classification decisions and construct a partition on the set of NPs with one cluster for each set of coreferent NPs.

As described below, knowledge-based approaches and machine learning approaches differ precisely with respect to how the prediction function is acquired.

1.4.2 Knowledge-Based Approaches

Knowledge-based approaches is a term used to refer to approaches that rely on hand-crafted heuristics for performing a task. In knowledge-based approaches to coreference resolution, the function for predicting whether two NPs are coreferent is composed of a set of hand-crafted heuristics. In essence, these heuristics encode coreference constraints and preferences, and are expressed in terms of the knowledge sources available to the classification procedure. Each constraint specifies whether two NPs must or must not be coreferent. On the other hand, each preference specifies whether NPs are likely to be coreferent or not.

1.4.3 Machine Learning Approaches

In contrast to knowledge-based approaches, *corpus-based approaches* or *machine learning approaches* acquire a prediction function *automatically* via the use of a

machine learning algorithm. In machine learning, a discrete-valued classification procedure is also known as a *classifier*. In other words, a classifier is a function $C : X \rightarrow Y$, where X is a set of objects to be classified, and Y is a set of possible labels of an object in X. For coreference resolution, each $x \in X$ is a description of two NPs, NP_i and NP_j, denoted by $inst_{(NP_i,NP_j)}$, where NP_i precedes NP_j in the associated text. The set Y simply consists of two labels, COREFERENT and NOT COREFERENT.

The particular learning paradigm we rely on in automatically acquiring a classifier is called *supervised learning*. In this learning setting, a learning algorithm is used to acquire a classifier from a set of *labeled examples/instances* L. Using the notation introduced above, $L = \{(x_i, y_i) \mid x_i \in X \text{ and } y_i \in Y\}$. In other words, each labeled example is an ordered pair (x, y) comprising an object to be classified (x) and its class label (y). Moreover, each object x is represented as a vector of *features*, which is a term specifically employed in the machine learning community to refer to the knowledge sources available to a learning algorithm. As mentioned above, these features or knowledge sources are a means of feeding problem-specific information to the learning algorithm and therefore are supposed to be useful for predicting the class label of a new example. Note that we only need to decide *what* features the learning algorithm has access to; the learning algorithm will decide *how* to use the given features. Now, given a *training set* consisting of labeled instances, a learning algorithm can automatically induce a classifier for predicting the label of a *test instance* (i.e., an unlabeled instance).

1.4.4 Why Machine Learning Approaches?

Perhaps a natural question to ask at this point is: why focus on machine learning approaches instead of knowledge-based approaches? Our decision essentially stems from the following observations.

Most importantly, there is empirical evidence that learning-based coreference systems can outperform their knowledge-based counterparts. For instance, both Aone and Bennett (1995) and McCarthy and Lehnert (1995) report that their learned coreference classifier outperform their rule-based resolver. More recently, Soon et al. (2001) have shown that their learning-based coreference system achieves performance comparable to state-of-the-art coreference systems, all of which are knowledge-based.

Moreover, as is clear from the discussion in the preceding subsection, machine learning approaches offer a number of advantages over knowledge-based approaches to coreference resolution. First, they can significantly reduce the amount of time and linguistic expertise required to construct a coreference ruleset by automating its acquisition directly from labeled data. In particular, the public availability of several coreference corpora, including the ones created for the Message Understanding Conference (MUC) (MUC-6, 1995; MUC-7, 1998) and the Automatic Content Evaluation (ACE) research program[5], has facilitated the application of machine learning approaches to the problem. Second, they allow coreference classifiers to be trained on the real-world text on which they will be used; the resulting classifiers, therefore, can potentially provide robustness and accuracy in the face of incomplete and and noisy input. Finally, they make it easy to empirically evalu-

[5]See `http://www.itl.nist.gov/iad/894.01/tests/ace` for details on the ACE research program.

ate the contribution of each knowledge source that is accessible to the coreference system.

1.5 Contributions

In this dissertation, we will extend the standard machine learning framework discussed the previous section for performing noun phrase coreference resolution. Like the standard framework, our machine learning framework retains the advantages of being *language-* and *corpus-independent*. In particular, although our experiments only handle coreference phenomena in *English*, this is a function of the features employed by the learning framework and not of the framework. Similarly, although we restrict our attention to coreference resolution of *noun phrases* in our experiments, this is a function of the data sets and again not of the framework. In other words, our framework should work across different natural languages and be able to handle different types of anaphora, provided that corpora annotated with the desired anaphora information from the language(s) of interest are available.

The contribution of the dissertation lies in generalizing and improving the standard machine learning approach to coreference resolution, as described below.

A best-performing coreference resolution system. We present a fully automatic, learning-based coreference resolution system that achieves the best results reported to date on the standard MUC-6 and MUC-7 coreference data sets. In addition, our system consistently outperforms the best previously existing learning-based coreference system — the Soon et al. (2001) coreference system — on both the MUC and the ACE coreference data sets.

Linguistic extensions to existing machine learning approaches to coreference resolution. We propose two types of linguistic modifications, as described below.

- **Large-scale expansion in the number and sophistication of coreference knowledge sources.** As discussed in Section 1.2, one of the difficulties of coreference resolution lies in its reliance on sophisticated knowledge sources. However, most existing learning-based coreference systems rely on a fairly small set of surface-level features. Though easily computable, these features are usually inadequate for handling the large variety of coreference relationships. Hence, we investigate a large-scale expansion in the number and sophistication of the features commonly employed in existing coreference resolution systems.

- **A new corpus-based approach to anaphoricity determination.** As mentioned in Section 1.1, we view coreference resolution essentially as anaphora resolution, in which the clustering algorithm is used to find an antecedent for each anaphoric noun phrase. Despite the potential usefulness of knowledge of anaphoricity for coreference resolution, to our knowledge none of the existing learning-based coreference systems makes use of such information. We present a new supervised approach to identifying anaphoric and non-anaphoric noun phrases and show how such anaphoricity information can be used to improve learning-based coreference systems.

Extra-linguistic modifications to existing learning approaches to coreference resolution. We propose two types of extra-linguistic modifications, as described below.

- **Error-driven rule pruning.** Recall that the standard machine learning approaches to coreference resolution combines classification and clustering. As seen in Section 1.4, recasting the problem as a classification task precludes enforcement of the transitivity constraint inherent in the coreference relation. Hence, the clustering mechanism is needed to coordinate these possibly contradictory pairwise classifications. In addition, because the coreference classifiers are trained independently of the clustering algorithm to be used, improvements in classification accuracy do not guarantee corresponding improvements in clustering-level accuracy. Therefore, to more tightly tie the classification- and clustering-level coreference decisions, we propose an error-driven rule pruning algorithm that optimizes the coreference classifier with respect to the clustering-level coreference scoring function.

- **Augmenting existing system components with alternative implementations.** Recall that, to implement the standard machine learning framework for coreference resolution, one has to specify the learning algorithm for training a coreference classifier, the clustering algorithm for coordinating classification decisions, and the method of creating training instances. On the other hand, our framework generalizes the standard framework by only requiring one to specify a *set* of learning algorithms, clustering algorithms, and training instance creation methods that are potentially useful for building a high-performing coreference engine at the time of system construction. In particular, our framework delays the decisions of which learning algorithm, clustering algorithm, and training instance creation method to use until training time. This makes it possible to base such design decisions on the data set on which the coreference system is trained, thus potentially

allowing the system better capture the specificities of the data set.

A data-driven approach to component selection for a learning-based coreference system. The above modifications are proposed based on our intuition and belief that they will contribute positively to the performance of a coreference resolution system. Even if each of the modifications is beneficial to the coreference system, it is possible that the modifications together may interact in such a way that produces an adverse effect on the performance of the system. In other words, each modification may be locally but not globally beneficial. Therefore, it is necessary to select a subset of modifications that, when applied in conjunction, can improve system performance. We propose a data-driven approach to component selection. Specifically, we exhaustively enumerate all possible subsets of modifications and select the one that achieves the "best" performance on held-out data according to some user-defined criteria.

Extensive performance analysis. We present a detailed analysis of the performance of our approach to coreference resolution. As we will see, our analysis not only allows us to gain additional insights into the linguistic and extra-linguistic aspects of coreference resolution, but also reveals what we need to do to obtain further performance improvements.

1.6 Roadmap

In this chapter, we first introduced the problem of noun phrase coreference and showed that it is a hard and important problem in NLP. Then we gave an overview of knowledge-based and machine learning approaches to coreference resolution and

explained our decision of focusing on machine learning approaches. Finally, we presented the contributions of this research.

The rest of the dissertation is structured as follows. Chapter 2 describes a generic algorithm that can be regarded as a skeleton behind the design and development of most contemporary anaphora and coreference resolution algorithms. In Chapter 3, we compare and contrast existing approaches to anaphora and coreference resolution in the context of the generic algorithm presented in Chapter 2. Chapter 4 presents our learning-based coreference resolution system, centering the discussion around our linguistic and extra-linguistic modifications to the standard machine learning framework. In Chapters 5 and 6, we present evaluation results of our coreference system and analyze its performance, respectively. Chapter 7 summarizes the contributions of this research, discusses our recent work on weakly supervised approaches (i.e., approaches that aim to reduce a learning algorithm's reliance on annotated data) to coreference resolution, and concludes with future directions.

CHAPTER 2

A GENERIC ALGORITHM FOR ANAPHORA AND COREFERENCE RESOLUTION

> *Dealing with anaphoric language can be decomposed into two complementary tasks: (1) identifying what a text potentially makes available for anaphoric reference and (2) constraining the candidate set of a given anaphoric expression down to one possible choice.* — Webber (1979)

In this chapter, we will introduce a generic algorithm, **GenericResolve**, for anaphora and coreference resolution. As we will see, the algorithm provides a framework for characterizing and analyzing existing resolution algorithms, thus enabling us to set the stage for the discussion of related work in the next chapter.

2.1 The Algorithm

As noted in Section 1.6, virtually all anaphora resolution algorithms and coreference resolution algorithms that view coreference as anaphora resolution are instantiations of **GenericResolve**. **GenericResolve**, shown in Figure 2.1, takes as input an unrestricted text D and searches for an antecedent for each potentially anaphoric discourse entity in D. The first three steps of the algorithm are performed at the document level, whereas the remaining steps are performed for each discourse entity in the document. A step by step explanation of the algorithm follows.

Step 1: Identification of discourse entities For pronoun resolution algorithms, the task here is simply to identify all of the pronouns in the associated

---

GenericResolve(D)

Input: An unannotated document D

Algorithm:

◁ 1. IDENTIFICATION of discourse entities from D ▷
Produce a list of discourse entities $E := \{\ e \mid e$ is a discourse entity that appears in $D\ \}$. Initialize the list of potentially anaphoric entities $A := E$.

◁ 2. CHARACTERIZATION of discourse entities ▷
Compute for each discourse entity $\text{NP}_i \in E$ a set of values $\{\ k_{i1},\ k_{i2},\ \ldots,\ k_{im}\ \}$ from m
knowledge sources.

◁ 3. ANAPHORICITY DETERMINATION (**Optional**) ▷
Compute the list $A := A \setminus \{\ e \in A \mid e$ is not anaphoric $\}$.

foreach NP_j in A **do**

◁ 4. GENERATION of candidate antecedents ▷
Compute the list of candidate antecedents of NP_j $C_j := \{\ e \in E \mid e$ lies within the scope of $\text{NP}_j\ \}$.

◁ 5. FILTERING (**Optional**) ▷
Compute the list $C_j := C_j \setminus \{\ e \in C_j \mid e$ satisfies a non-coreferent constraint with $\text{NP}_j\ \}$.

◁ 6. SCORING/RANKING (**Optional**) ▷
Score or rank each NP in C_j and sort C_j w.r.t. the score.

◁ 7. SEARCHING/CLUSTERING ▷
Select an antecedent for NP_j from C_j and annotate the document accordingly.

endfor

Output: A document D annotated with anaphoric information

Figure 2.1: The **GenericResolve** Algorithm. In this algorithm, a list is viewed as a set whose elements can be ordered and hence all basic set operations can be applied.

text. For coreference resolution algorithms, the task is to identify all of the noun phrases in the text.

Step 2: Characterization of discourse entities This step is composed of two sub-tasks. The first task is to define a representation of a discourse entity, or equivalently, a way to characterize a discourse entity. The representation precisely determines the knowledge regarding a discourse entity that the algorithm needs in order to perform the remaining steps successfully. Once a representation is created, the second task is then to instantiate the representation for each entity.

Step 3: Anaphoricity determination The goal of ANAPHORICITY DETERMINATION is to determine whether a discourse entity is anaphoric or not. Non-anaphoric entities, by definition, do not possess an antecedent and hence the algorithm will not need to search for an antecedent for these entities if anaphoricity information is available. Some algorithms do not perform anaphoricity determination, in which case all discourse entities are implicitly assumed to be potentially anaphoric.

Step 4: Generation of candidate antecedents Unlike the previous steps, this step and the remaining ones are performed for each anaphoric entity in the document. Once an NP is determined to be anaphoric, the algorithm identifies the scope of the NP and gathers those NPs that fall within its scope to generate the list of candidate antecedents for it. For most anaphora and coreference resolution algorithms, the scope of the anaphor is not identified for computational reasons such as complexity and accuracy, and the list of candidate antecedents in this case is simply taken to be the list of NPs preceding the anaphor in the associated document.

Step 5: Filtering FILTERING involves removing unreasonable candidate antecedents for each anaphoric noun phrase. Specifically, FILTERING is performed based on a set of rules or hard constraints that specify non-coreference or contra-indexing between two discourse entities. A potential antecedent that satisfies any of the non-coreferent constraints can therefore be removed from the candidate list. This step aims at reducing the amount of processing that needs to be performed by the algorithm, and can potentially improve the accuracy of the resolution procedure.

Step 6: Scoring/Ranking SCORING, or RANKING, of the remaining candidates is performed based on a set of rules or soft constraints that indicate preferences that two NPs co-specify. In some cases, each candidate antecedent for a given anaphor is assigned a numeric score that reflects the likelihood that the two NPs under consideration possess an anaphoric or coreferent relationship, and the list of antecedents is subsequently sorted based on the scores. In other cases, only a ranking of the candidate antecedents is generated based on a set of rules or discourse principles. Overall, the goal of this step is to rank candidate antecedents according to the preference adopted by a specific algorithm.

Step 7: Searching/Clustering The goal of this step is to select an antecedent for a given anaphor from the list of candidate antecedents returned by the previous step. If this list is empty, then no antecedent will be selected for the anaphor. If SCORING or RANKING of the candidate antecedents has been performed by step 6, which is optional, then SEARCHING becomes the trivial task of selecting the first (or highest-ranking) element in the candidate list as the antecedent of the anaphor.

Otherwise, this list of candidate antecedents is searched in some order specified by the resolution algorithm, and the "best" NP encountered is then selected as the antecedent. In the case of coreference resolution, this process is essentially equivalent to applying a single-link clustering algorithm to each anaphoric NP to cluster the NPs in the document and generate a partition on them.

Note, however, that steps 3, 5 and 6 can be absent in an anaphora or coreference resolution algorithm, although it is not common to have an algorithm in which all three steps are omitted. Existing resolution algorithms differ in the way these seven steps are implemented. As a result, the generic algorithm can be thought of as providing a natural way of characterizing resolution algorithms. We will discuss this issue in detail in the next chapter.

2.2 Chapter Summary

In this chapter, we have described a generic algorithm, **GenericResolve**, for anaphora and coreference resolution. In particular, existing anaphora and coreference resolution algorithms can all be viewed as instantiations of **GenericResolve**, differing primarily in the way each of the seven steps of the algorithm is implemented. In the next chapter, we will discuss related work on computational approaches to anaphora and coreference resolution in the context of this generic algorithm.

CHAPTER 3

RELATED WORK

Many AI workers, myself included, adhere to the maxim "One good theory is worth a thousand heuristics". — Hirst (1981)

In this chapter, we will attempt to characterize existing computational approaches to anaphora and coreference resolution in the context of the **GenericResolve** algorithm presented in the previous chapter. We will begin with a discussion of several trends that are representative of ongoing research in this area. As each trend normally signifies a shift in the overall research direction, the discussion here will set the stage for comparison between different approaches in the subsequent subsections.

3.1 Research Trends

Research on anaphora and coreference resolution in computational linguistics has proceeded in various directions since its inception three decades ago. Nevertheless, we are able to observe the following research trends with respect to this problem.

From knowledge-rich approaches to knowledge-lean approaches. Anaphora and coreference resolution systems differ with respect to the *types* of knowledge sources required to perform accurate resolution. This difference separates *knowledge-rich* systems — which usually require domain-specific knowledge, semantic and discourse analysis, as well as sophisticated inference mechanisms — from *knowledge-lean* systems, which deliberately avoid the use of sophisticated knowledge and complex analyses in the resolution procedure and instead rely only

on morphological and possibly (shallow) syntactic information. While early research on this problem focuses on knowledge-rich approaches, the need for fast and robust solutions to anaphora and coreference resolution has prompted the development of knowledge-lean approaches (Mitkov et al., 2001).

From semi-automatic pre-processing to fully automatic pre-processing. An anaphora or coreference resolution system takes a raw text, not an annotated text, as input. Consequently, an input text must be pre-processed before any given resolution procedure can be applied to it. The first two steps in **GenericResolve** essentially outline the common pre-processing steps: IDENTIFICATION and CHARACTERIZATION of discourse entities. Specifically, a coreference system needs to identify all noun phrases from an input text. During CHARACTERIZATION, knowledge required by the resolution algorithm is computed for each discourse entity identified in the first step. For instance, *gender* and *number* are two common knowledge sources employed by resolution systems. Neither NP identification nor knowledge computation can be achieved with perfect accuracy, however. Early research in anaphora and coreference resolution usually assumes that pre-processing is error-free. This entails manual fixing of errors made by the pre-processing routines. On the other hand, the focus of recent research has shifted towards the construction of fully automatic anaphora and coreference resolution systems in which the entire process is completely free from human intervention.

From small-scale evaluation to large-scale evaluation. Evaluation of early anaphora resolution algorithms is usually performed by hand. In fact, some of these algorithms were not implemented as computer programs at the time of evaluation. Since the evaluation process was not automatic, these algorithms were mostly

evaluated on a relatively small number of unseen cases. The recent availability of a huge number of online texts as well as the development of fairly robust pre-processing routines such as morphological analyzers and shallow parsers have aided the automation of anaphora resolution systems and made automatic, large-scale evaluation of resolution algorithms possible.

From knowledge-based approaches to knowledge-lean approaches. Early anaphora resolution systems adopt knowledge-based approaches, in which the resolution procedure is based on a set of hand-crafted heuristics that specify whether two discourse entities can or cannot have an anaphoric relationship. Owing to the availability of corpora annotated with coreference information and the success of data-driven approaches in many NLP tasks, researchers in recent years have attempted to apply corpus-based/empirical methods to anaphora and coreference resolution, in which anaphora or coreference knowledge is derived automatically from labeled data.

In principle, we can characterize existing work on anaphora and coreference resolution along any of the above dimensions. Nevertheless, we will mainly focus our discussion on the last dimension, knowledge-based versus corpus-based approaches (Sections 3.2 and 3.3), for the following reasons. First, as we will be discussing potential problems with existing machine learning approaches to coreference resolution in Chapter 4, a characterization of related work along the last dimension seems most pertinent to the subsequent discussion. More importantly, the trend along the other dimensions typically follows from the last dimension — corpus-based methods allow large-scale evaluations of coreference systems that employ fully automatic pre-processing and knowledge-lean methods. Alternatively,

the other trends seem to go together. In most cases, for example, large-scale evaluations are possible only if fully automatic pre-processing is employed. In spite of the relationship that exists among the dimensions, we will briefly discuss the remaining three dimensions in Sections 3.4, 3.5 and 3.6.

3.2 Knowledge-Based Approaches

As we mentioned before, virtually all anaphora and coreference resolution algorithms are instantiations of the generic algorithm described in Chapter 2. As anaphora and coreference resolution systems adopting knowledge-based approaches differ from those adopting corpus-based methods primarily with respect to the way the FILTERING step and the SCORING step are implemented, the comparison between the various approaches will be based mainly on these two steps. In particular, IDENTIFICATION of discourse entities is trivial for pronoun resolution and is performed similarly among coreference resolution systems using a noun phrase finder. In addition, GENERATION of candidate antecedents for a potential anaphor in most systems simply involves the construction of a set of noun phrases preceding the anaphor under consideration in the associated document, although some systems impose a limit on how far apart an anaphor and a candidate antecedent can be. Consequently, these two steps will not be discussed explicitly, although other steps will be mentioned as necessary. In the rest of this subsection and the next, we will discuss related work using knowledge-based approaches and corpus-based approaches respectively.[1]

For systems adopting knowledge-based approaches, the rulesets used in FIL-

[1]A more detailed description and complementary analysis of most of the systems discussed below can be found in Mitkov (1999).

TERING and SCORING are composed entirely of hand-coded constraints. As noted above, FILTERING involves removing any candidate antecedent that violates any of the non-coreferent constraints from the candidate list. Some commonly pre-specified constraints employed by knowledge-based systems are traditional linguistic constraints such as agreement constraints. In addition, the score associated with each soft constraint used in the SCORING step of knowledge-based systems is also pre-specified rather than learned. Broadly speaking, knowledge-based systems can further be divided into three classes: syntax-based approaches, discourse-based approaches, and factor-based approaches.

3.2.1 Syntax-Based Approaches

Syntax-based anaphora resolution algorithms rely solely on syntactic and morphological information and assume the existence of a full parse tree for each sentence in the input text. Both FILTERING and RANKING occur during the SEARCHING step. Specifically, for each potential anaphor, the search for an antecedent is performed via the traversal of parse trees. When a candidate antecedent is encountered and it does not survive the FILTERING step, the search for the next candidate continues. Otherwise, the candidate is selected as the antecedent. RANKING of candidate antecedents is encoded implicitly in the order in which the tree nodes (and hence the candidates) are visited.

Hobbs' (1978) naive algorithm is one of the most well-known syntax-based algorithms for resolving pronoun references (Tetreault, 1999). Specifically, the algorithm considers the sentences in the text in reverse order, starting from the sentence in which the pronoun resides and searching for potential antecedents in the corresponding parse trees in a left-to-right, breadth-first manner that obey binding

and agreement constraints. The algorithm's preferences for recency (i.e., NPs that are in closer proximity to the anaphor) as well as for NPs in the subject position are generally believed to be the reason for its good performance on pronouns with intra-sentential antecedents (Lappin and Leass, 1994; Walker, 1989). Despite its simplicity, the algorithm has received criticisms regarding its dependence on full parse trees (Ge et al., 1998).

3.2.2 Discourse-Based Approaches

Discourse-based methods for anaphora resolution have been developed out of the belief that some types of anaphora can only be resolved with an understanding of the focus (or center) of an utterance.[2] For instance, consider the following sentence.

William saw Harry in the park today. He was talking to his friend.

Evidently, the resolution of the pronoun *He* cannot be based solely on grammatical information such as gender and number, since both *William* and *Harry* are potential antecedents of *He*. Nevertheless, there is an indication that the center of the discourse has shifted from *William* to *Harry* in the first sentence. Consequently, most readers would perceive *He* as referring to *Harry* and not *William*, since pronominalization serves to capture attention and hence the noun phrase that is the focus at a particular point in the discourse is more likely to be pronominalized than the other NPs.

Discourse-based anaphora resolution algorithms rely on principles such as coherence and focusing to track the center of an utterance. Discourse-level constraints

[2]Grosz et al. (1995) define an utterance as "the uttering of a sequence of words at a certain point in a discourse". They emphasize that "it is an utterance and not a sentence in isolation that has centers".

and preferences formulated from discourse principles are then incorporated into the FILTERING step and the RANKING step of the resolution procedure respectively, in addition to syntactic and morphological constraints that are commonly applied in these steps by other types of anaphora resolution algorithms. Algorithms in this category differ from those in other categories primarily with respect to the way RANKING is performed: for discourse-based algorithms, ranking of candidate antecedents is driven predominantly by discourse principles, as opposed to syntax-based methods which do not perform any ranking and factor-based methods which score candidate antecedents based on a number of factors. Below we subcategorize discourse-based resolution algorithms. We will first introduce algorithms that are built upon centering theory. Then we will discuss algorithms that are based on a more general notion of focus.

Centering Theory and Centering Algorithms

Grosz and Sidner (1986) develop a theory of discourse structure, and the part of the theory that has sparked a lot of research on focus-based anaphora resolution algorithms is *centering* (Grosz et al., 1995). Centering theory models the local coherence of a discourse, and is composed of a set of constraints governing center movement (the conditions under which the center of an utterance can or should move from one discourse entity to another) and center realization (the conditions under which a discourse entity can be pronominalized) that are formulated based on psycholinguistic and cross-linguistic evidence. Generally speaking, there are two principles underlying centering theory: cohesion (maintaining the same center) and salience (realizing the center as the most prominent NP). Note, however, that centering is a theory for interpreting pronouns in a discourse and its use in anaphora

and coreference resolution is only one of the applications of the theory. Anaphora resolution algorithms that are built upon centering constraints are collectively known as *centering algorithms*.

The BFP centering algorithm. One of the most well-studied centering algorithms is the BFP algorithm (Brennan et al., 1987). BFP can also be viewed as an instantiation of **GenericResolve**. Specifically, FILTERING of candidate antecedents for a given anaphor in BFP makes use of syntactic constraints such as binding constraints and morphological constraints such as gender and number. The RANKING of candidate antecedents, however, is guided by centering principles.

Improvements to BFP. A series of studies (Kehler, 1997a; Walker, 1989; Tetreault, 2001) reveals several weaknesses of the BFP algorithm, some of which are also exhibited by other centering algorithms. First, the algorithm is weak in resolving pronouns that refer to a global focus, since centering only captures local discourse phenomena. Second, BFP does not have an intrasentential processing mechanism, since the algorithm attempts to search for antecedents for a given pronoun only in the previous utterances. Finally, the ranking of candidate antecedents is not completely specified by the algorithm, which means that it is possible for two candidates to have the same ranking.

There have been various proposals that attempt to remedy the above problems. For instance, Walker (1989) describes a way of incorporating a global focus in the centering framework. Tetreault (2001) proposes a new centering algorithm, Left-Right Centering (LRC), that can be viewed as extending the Hobbs algorithm with discourse information to direct the search for antecedents and therefore can resolve pronouns incrementally. Furthermore, Kameyama describes a method, which she

calls *intrasentential centering*, to enhance BFP with an intrasentential processing mechanism (Kameyama, 1998). Finally, if potential antecedents for an anaphor can be proposed one after the other as in the Hobbs algorithm, there would be no ambiguity of preference for antecedents.

Focus-Based Approaches

The role of focus in anaphora resolution was recognized even before centering theory was developed. Unlike centering algorithms, some of the focus-based algorithms described here are designed to resolve definite noun phrases rather than pronouns.

Grosz's stack-based approach. Grosz (1977), for instance, defines a representation, *focus space*, for tracking the focus of a discourse in task-oriented dialogs. More precisely, each focus space comprises all entities in the same discourse segment. An *explicit focus space* contains all discourse entities explicitly mentioned in the dialog, whereas an *implicit focus space* contains all entities that are either related to explicitly mentioned entities or implicitly referred to in the dialog. The current discourse segment, or the *active focus space*, is on the top of the focus stack, and a list of *open focus spaces* that consist of entities that might be referred to later in the dialog is kept in the stack. The active focus space is popped off the stack when there is a focus shift. The discourse entities in each focus space are represented in an elaborate semantic network. When attempting to resolve a definite NP during SEARCHING, the search procedure is constrained by considering a "focused match" only between the network in which the definite NP resides and the network representing the active focus space. Consequently, the candidate list

is composed exactly of the entities in the active focus space, and NPs residing in other focus spaces are effectively filtered from the list. There are two problems associated with Grosz's approach, however. First, the approach can only handle definite descriptions but not pronominal references. In addition, the approach seems to be applicable only to task-oriented dialogs. It is unclear how the approach can be extended to other other genres where it may be harder to detect a shift in focus.

Sidner's mutual bootstrapping method for anaphora resolution. Sidner (1979, 1981) develops a local focusing framework that extends Grosz's approach to enable the resolution of pronominal references and recognize focus shifts in broader situations. She argues that the two processes, *focus tracking* and *anaphora resolution*, influence each other, and her framework comprises two algorithms, the *focusing* algorithm and the *anaphora resolution* algorithm, that reflect this circular dependency. To represent the current state of a discourse, Sidner's framework maintains a set of data structures, including the *current focus*, the *actor focus* that is used to handle agentive pronouns specifically, a list of *alternative candidate foci*, and a *focus stack*. For each sentence, the focusing algorithm uses a set of rules as well as results from anaphor interpretation to determine whether there is a shift in focus and updates the data structures accordingly. Then, for each anaphor encountered, the anaphora resolution algorithm uses another set of rules to rank candidate antecedents based on the focus-tracking data structures. Since the algorithm only considers NPs in these data structures as candidate antecedents, implicit FILTERING of discourse entities occurs when the focusing algorithm updates the data structures.

Azzam's extension to Sidner's algorithm. Azzam (1996) points out two problems with Sidner's approach. First, Sidner's algorithm is designed to handle sentences with relatively simple structures, i.e., sentences of the form *Subject Verb Object* optionally followed by prepositional phrases and adverbial adjuncts. A complex sentence must be segmented into simpler constituents before her algorithm can be applied, but Sidner does not provide any mechanism for doing so. Second, the focus-tracking data structures do not contain entities from the sentence under consideration. Since the anaphora resolution algorithm proposes only candidate antecedents residing in these data structures, intrasentential antecedents are virtually impossible. Azzam thus proposes a simple extension to Sidner's framework that handles both problems simultaneously : making embedded sentences (i.e., sentences that correspond to elementary events) the unit of discourse processing. She then presents an algorithm for extracting embedded sentences from complex sentences and applies Sidner's algorithm to embedded sentences instead. For evaluation, she extends the coreference resolution algorithm used in the LaSIE information extraction system (Gaizauskas et al., 1995) with her focusing mechanism, but concludes that there is no observable difference between the performance of the coreference algorithms with and without focusing (Azzam et al., 1998).

Strube's S-list algorithm. Strube (1998) proposes a new pronoun resolution algorithm that can be viewed as an alternative to centering. Specifically, a simple data structure, S-list, is used to store all discourse entities in the previous and current utterances, effectively FILTERING or eliminating all NPs that do not exist in the previous or current utterance from consideration. SCORING occurs immediately after each discourse entity encountered is added to the list, which is sorted such that

its elements are always in decreasing salience with respect to the current discourse. The way salience is computed roughly corresponds to Prince's *familiarity scale* (Prince, 1981). SEARCHING proceeds simply by selecting the first element in the list that satisfies the agreement constraints with the pronoun.

3.2.3 Factor-Based Approaches

Factor-based anaphora resolution algorithms seek to combine various knowledge sources, including morphological, syntactic, semantic, and in some cases pragmatic information, for antecedent selection. These "factors" can be broadly divided into two categories: constraints (that must be satisfied) and preferences (that distinguish between candidates satisfying all constraints). In contrast to discourse-based methods, factor-based methods do not rely on an elaborate discourse theory that usually involves complex modeling of attentional states, and antecedent selection is not dominated by discourse principles, which have an intrinsic preference for intersentential antecedents (Walker, 1989). In particular, discourse information in this class of algorithms is formulated as preferences rather than constraints.

When viewed as instantiations of **GenericResolve**, factor-based anaphora resolution algorithms operate as follows: To resolve a potentially anaphoric noun phrase, constraints are first applied to narrow down the set of potential antecedents during the FILTERING step. Subsequently, each preference factor (independently) proposes an antecedent for the noun phrase under consideration during the RANKING/SCORING step. Strictly speaking, each preference factor represents a constraint that can either be hard (in which case only one antecedent is selected) or soft (in which case a numeric score is assigned to each candidate antecedent reflecting the preference for antecedence). The possibly contradictory individual

decisions are then coordinated and combined into a global decision via the use of a predefined metric. The SEARCHING step then relies on the global decision reached for each candidate antecedent to select the antecedent for the anaphoric noun phrase. Despite these similarities, algorithms in this class differ with respect to the *linguistic knowledge sources* being used as well as the *strategies* being employed to combine these knowledge sources. Although existing work on anaphora and coreference resolution has generally focused on investigating the knowledge sources that are useful for the task, Mitkov (1997) shows that extra-linguistic strategies (e.g., a traditional approach in which "bad" candidate antecedents are filtered before they are ranked versus an uncertainty-based approach in which *all* potential antecedents (i.e., no filtering) are assigned a score based on each available knowledge source) also play an important role in the performance of an anaphora resolution algorithm. As a result, we will discuss this class of resolution algorithms in terms of both the knowledge sources and the strategies used to combine them in the rest of the subsection.

Carbonell and Brown's multi-strategy approach. Carbonell and Brown (1988) develop an anaphora resolution algorithm under the hypothesis that anaphora resolution can be best accomplished via "a combination of a set of strategies rather than by a single monolithic method". Various constraints for anaphora resolution are proposed: syntactic (e.g., gender and number), semantic (e.g., selectional constraints), and pragmatic (e.g., eliminate a candidate antecedent from consideration if some action that occurs between the antecedent and the anaphor implies that the two cannot possess an anaphoric relationship). As far as preferences are concerned, factors such as recency, topicalization (i.e., entities that

appear in topicalized structures), syntactic parallelism (i.e., entities having the same grammatical role as the anaphor) and semantic parallelism (i.e., entities having the same thematic role as the anaphor) are employed for antecedent selection. Knowledge sources are combined to select an antecedent for an anaphor as follows: candidate antecedents that violate any of the constraints are filtered out. If more than one candidate is left, the preferences are employed. If more than one preference applies and each applicable preference proposes a different antecedent, then the anaphor is considered to have a truly ambiguous antecedent.

Lappin and Leass's syntax-based approach. Lappin and Leass (1994) propose a pronoun resolution algorithm that relies on a set of syntax-based constraints and salience-based preferences. In contrast to Carbonell and Brown (1988) who make use of semantic and pragmatic constraints that are generally hard to encode with a reasonable accuracy, Lappin and Leass instead employ only morphological constraints such as gender and number and syntactic constraints such as binding constraints. Additionally, as opposed to most previous work in which pleonastic pronouns are manually filtered out during evaluation, Lappin and Leass design a separate set of patterns to identify pleonastic pronouns explicitly. During SCORING, an anaphoric entity is resolved to the most salient preceding entity, where the salience of a noun phrase is given by the salience of the coreference equivalence class to which the noun phrase belongs.[3] The salience of a class is in turn calculated based on a set of *salient factors* applied to each member of the class. Each salience factor (attached to an NP) is associated with an initial weight that

[3]Here, coreference equivalence classes are dynamically constructed during the resolution process in which an anaphoric pronoun is assigned to the same class as the discourse entity to which it is resolved. Nevertheless, full coreference is *not* assumed.

indicates the relative contribution of the factor to overall salience of the NP. As these initial weights are reduced after each utterance in the discourse is processed, the salience of each equivalence class constructed so far needs to be re-calculated accordingly. The salience factors are fairly intuitive. For example, both sentence recency and the grammatical role of a discourse entity play a role in determining the salience.

Kennedy and Boguraev's parser-free pronoun resolution system. One advantage of Lappin and Leass's algorithm is that it does not depend on semantic or pragmatic information for pronoun resolution. However, the availability of perfect parse trees is too strong an assumption to make in realistic settings. To further relax this assumption, Kennedy and Boguraev (1996) propose an algorithm that is structurally equivalent to that of Lappin and Leass except that the algorithm does not rely on deep syntactic parsing. Specifically, the information provided by the full parser in Lappin and Leass's algorithm (such as the grammatical role of an NP) is approximated by a set of ten heuristics in Kennedy and Boguraev's algorithm.

Baldwin's high-precision pronoun resolution system. Baldwin (1997) describes a high-precision rule-based pronoun resolution algorithm, CogNIAC. Like Kennedy and Boguraev's algorithm, CogNIAC only has access to morphological information and shallow syntactic information that is heuristically computed. The algorithm is composed of a set of rules that aims to recognize anaphoric relationships. The rules are ordered according to precision and importance with respect to the pronoun resolution task and are successively applied to the pronoun under consideration. The resolution of a pronoun terminates as soon as a rule applies to

the pronoun.

Mitkov's knowledge-lean robust pronoun resolution system. Mitkov (1998) acknowledges the importance of developing knowledge-lean approaches to anaphora resolution and proposes a robust algorithm for resolving pronouns in technical manuals. Like CogNIAC, Mitkov's algorithm relies solely on morphological and shallow syntactic information provided by a part-of-speech tagger and a base noun phrase finder respectively. During FILTERING, any potential antecedent that either violates agreement constraints or is more than two sentences away from the pronoun under consideration is removed from the candidate list. Each remaining candidate is scored based on a set of antecedent indicators, each of which potentially affects the salience of an NP and effectively serves as a preference for antecedent selection. A positive score is assigned to a candidate by a boosting indicator and a negative score assigned by an impeding indicator. The candidate with the highest aggregate score is then proposed as the antecedent. While the algorithm has been applied only to technical manuals, Mitkov argues that the majority of the antecedent indicators appear to be genre-independent. Unlike most previous work in which the salience of an NP is dependent almost exclusively on positional information (e.g., recency) and grammatical role, some of Mitkov's antecedent indicators make use of lexical information and NP type as additional knowledge sources. For example, impeding indicators are applicable not only to NPs embedded in prepositional phrases but also to indefinite NPs, and boosting indicators favor lexically reiterated items in addition to NPs appearing in section headings. Finally, the genre-specific indicators that the algorithm uses include preferences for NPs that represent terms used in the genre as well as those

that are objects of certain "indicating verbs".

Cardie and Wagstaff's clustering approach. Cardie and Wagstaff (1999) explicitly view coreference as a clustering task and use a right-to-left single-link clustering algorithm to coordinate the application of constraints and preferences for partitioning the given noun phrases into coreference equivalence classes. Specifically, they define a distance metric between two noun phrases that is essentially a linear combination of the (in)compatibility scores computed from the (relational) constraints and preferences. During CLUSTERING, if the distance between two noun phrases is less than the predefined clustering radius, then the corresponding equivalence classes are considered for possible merging. A wide variety of knowledge sources (encoded as features) are used by the clustering algorithm: lexical (e.g., head noun match, word overlap), syntactic (e.g., gender, number, animacy, apposition), semantic (e.g., WordNet class compatibilities), and positional (e.g., number of intervening noun phrases between the two NPs under consideration). Like Mitkov's system, the weight associated with each knowledge source in Cardie and Wagstaff's distance metric is chosen by hand rather than determined empirically.

In all of the factor-based approaches (except Baldwin (1997)) discussed in this subsection, the relative importance of each knowledge source to overall performance is not evaluated. As we will see in the next section, one potential advantage of corpus-based approaches over knowledge-based approaches is that the information regarding the relative importance of the factors comes directly with the induction of certain types of models (e.g., decision trees).

3.3 Corpus-Based Approaches

In contrast to knowledge-based approaches, corpus-based approaches rely on soft constraints that are derived from a corpus annotated with anaphora or coreference information in FILTERING and SCORING. For anaphora resolution, a model that determines the probability that an NP is an antecedent of a given anaphor is learned from a corpus. For coreference resolution, however, a model that determines the probability that two NPs are coreferent is learned instead; in addition, a separate CLUSTERING mechanism is needed to coordinate the possibly contradictory pairwise classifications and construct a partition on the set of NPs with one cluster for each set of coreferent NPs.

Another major distinction between corpus-based approaches and knowledge-based approaches is the need to create training instances for acquiring an anaphora/coreference model in the former. Training instances are typically created by relying on anaphora or coreference information from the training documents. The feature vector representing an instance essentially describes the two NPs involved and the context in which they occur, comprising relational features (i.e., features that test whether some property holds for the NP pair under consideration) and non-relational features (i.e., features that test some property of one of the NPs under consideration).

Corpus-based approaches differ from each other in terms of how the probabilistic model is learned and can further be divided into three classes: manual approaches, machine learning approaches, and statistical approaches. We will elaborate on each of these approaches in the following subsections.

3.3.1 Manual Approaches

Coreference systems in this class constitute a special case of corpus-based approaches in which the probabilistic model (which in this case is composed of a set of coreference rules) is mined from the corpus entirely by hand. Consequently, manual approaches are not widely adopted, since a lot of human intervention is required to derive the desired coreference information.

Harabagiu et al.'s knowledge-minimalist approach. Harabagiu et al. (2001) present a knowledge-minimalist approach for mining "easy" coreference rules from an annotated corpus. The goal is to acquire a set of (positive) rules for determining when two NPs are coreferent. The approach is based on the following observations:

- For any anaphoric noun phrase in a coreference chain[4], a positive training instance can be formed from the noun phrase and each of its preceding noun phrases in the same chain.
- One antecedent is sufficient for resolving an anaphor.
- Some coreference relationships require more knowledge and hence are harder to resolve than the others.

The first observation implies that more than one positive instance can be created from each anaphoric NP. The second observation implies that in principle, given n anaphors in the training set, we only need n positive instances to acquire the rules. These two observations together imply that we have the freedom to select positive instances from which to induce coreference rules, since the number of

[4]As we mentioned in Section 1.1, we assume throughout that a noun phrase is anaphoric if it is part of a coreference chain but is not the head of the chain.

available positive instances is at least as large as the minimum number of positive instances needed to derive the rules. The third observation implies that intelligent selection of "easy" positive instances can potentially minimize the knowledge needed to perform coreference resolution. The Harabagiu et al. algorithm first mines a set of rules for covering easy instances that are heuristically selected. The set of coreference rules is then transformed into a corresponding set of soft constraints by estimating the accuracy of each rule on the training data. To generate a partition on a given set of noun phrases from an unseen text, the algorithm relies on a local-search algorithm to search for the best possible partition. Specifically, the local-search algorithm starts with a random partition, makes local modifications to the partition, and picks the best partition at each iteration, where the score of a partition is computed from the pairwise coreference probabilities given by the soft constraints. The coreference system achieves a very high accuracy (approximately 90%) when evaluated on the MUC-6 coreference data set. Nevertheless, the system makes the strong assumption that all and only the NPs involved in coreference relationships are presented to the system for analysis. It is unclear whether the techniques would work well in the more realistic setting in which the set of NPs extracted by a shallow parser (which is presumably much larger than the set of NPs involved in non-singleton coreference classes) is analyzed by the coreference system instead.

3.3.2 Machine Learning Approaches

Unlike manual approaches, machine learning approaches to coreference resolution induce a model that determines the probability that two NPs are coreferent from annotated data *automatically* via the use of learning algorithms and can be charac-

terized in terms of the *knowledge sources* being employed, the method of *training data creation*, as well as the *learning algorithm* and the *clustering algorithm* being chosen.

Aone and Bennett's knowledge-rich anaphora resolution system for Japanese. Aone and Bennett (1995) describe a trainable system for classifying different types of anaphora occurring in Japanese texts about joint ventures. Specifically, the instance representation consists of 66 features, including lexical (e.g., part-of-speech), syntactic (e.g., grammatical role), semantic (e.g., semantic class), and positional (e.g., distance between the potential antecedent and the anaphor) features. Two different methods are used to create positive training instances: *transitive* (i.e., an instance is formed between an NP and each of its preceding NPs in the same anaphoric chain) and *non-transitive* (i.e., an instance is formed between an NP and its closest preceding NP in the same anaphoric chain). Negative instances are generated by pairing an NP with each preceding NP that does not have an anaphoric relationship with it. The system then uses the C4.5 decision tree induction system (Quinlan, 1993) to train an anaphora classifier that determines whether two NPs possess an anaphoric relationship. A best-first single-link clustering algorithm is then used to generate a partition on the set of NPs (see Figure 3.1). In essence, when an anaphor has more than one antecedent according to the coreference classifier, this clustering algorithm selects the one that is associated with the highest confidence value. Results show that the learned anaphora classifier achieves superior performance than the manually designed resolver.

McCarthy and Lehnert's domain-specific coreference system. McCarthy and Lehnert (1995) describe RESOLVE, a coreference resolution system that is in-

BestFirstClustering(NP_1, NP_2, ..., NP_k)

Input: A set of noun phrases $\{\text{NP}_1, \text{NP}_2, \ldots, \text{NP}_k\}$ in a test document

Algorithm:
Mark each NP_j as belonging to its own class: $\text{NP}_j \in c_j$.

foreach NP_j **do**
  Create a test instance, $inst(\text{NP}_i, \text{NP}_j)$, for all of its preceding NPs, NP_i.
  if $\exists\ i' < j$ such that $class(inst(\text{NP}_{i'},\text{NP}_j)) = \text{COREFERENT}$ **then**
    NP_{i^*} := *highest-confidence* preceding coreferent NP
    Merge c_{i^*} and c_j, where $\text{NP}_{i^*} \in c_{i^*}$ and $\text{NP}_j \in c_j$
  endif
endfor

Output: A partitioning on the noun phrases NP_1, NP_2, ..., NP_k.

Figure 3.1: The Best-First Clustering Algorithm

tended for use within an information extraction system. As a system that classifies coreferent phrases in the domain of joint ventures, 3 of the 8 features being employed are domain-specific. For example, there are features that test whether each of the NPs in the pair refers to a joint venture company (i.e., a company that is formed because of a tie-up among two or more entities). The domain-independent features can be characterized as lexical (e.g., whether the two NPs share a common noun phrase), semantic (e.g., whether one NP is an alias of the other), and positional (e.g., whether the two NPs are in the same sentence). No syntactic features are used. To generate positive training instances from coreference chains, only the *transitive* method is used. Negative training instances are generated by pairing an NP with each of its preceding non-coreferent NPs. An "aggressive-merge" clustering algorithm, shown in Figure 3.2, is used to coordinate the pairwise classification

AggressiveMergeClustering(NP_1, NP_2, ..., NP_k)

Input: A set of noun phrases {NP_1, NP_2, ..., NP_k} in a test document

Algorithm:
Mark each NP_j as belonging to its own class $\text{NP}_j \in c_j$.

foreach NP_j **do**
  Create a test instance, $inst(\text{NP}_i, \text{NP}_j)$, for all of its preceding NPs, NP_i.
  foreach $\text{NP}_{i'}$ such that $class(inst(\text{NP}_{i'},\text{NP}_j)) = \text{COREFERENT}$ **do**
    Merge $c_{i'}$ and c_j, where $\text{NP}_{i'} \in c_{i'}$ and $\text{NP}_j \in c_j$.
  endfor
endfor

Output: A partitioning on the noun phrases NP_1, NP_2, ..., NP_k.

Figure 3.2: The Aggressive-Merge Clustering Algorithm

decisions. Basically, each anaphoric NP is merged with all of its preceding NPs that are classified as coreferent with it. In other words, positive classifications are given a higher precedence than negative classifications when conflicts arise. Finally, McCarthy and Lehnert report that the learned coreference classifier outperforms a rule-based coreference classifier, a result that is consistent with that of Aone and Bennett.

Soon et al.'s knowledge-lean coreference system. Soon et al. (2001) adopt a knowledge-lean approach to general-purpose coreference resolution in which the decision tree learning algorithm has access to only 12 surface-level features. The features are all designed to be domain-independent, including one lexical feature (string matching), eight grammatical features (gender and number agreement, apposition, and NP types), two semantic features (semantic class compatibility and

ClosestFirstClustering(NP_1, NP_2, ..., NP_k)

Input: A set of noun phrases {NP_1, NP_2, ..., NP_k} in a test document

Algorithm:
Mark each NP_j as belonging to its own class: $\text{NP}_j \in c_j$.

foreach NP_j **do**
  Create a test instance, $inst(\text{NP}_i, \text{NP}_j)$, for all of its preceding NPs, NP_i.
  if $\exists\ i' < j$ such that $class(inst(\text{NP}_{i'},\text{NP}_j)) = \text{COREFERENT}$ **then**
    NP_{i^*} := *closest* preceding coreferent NP
    Merge c_{i^*} and c_j, where $\text{NP}_{i^*} \in c_{i^*}$ and $\text{NP}_j \in c_j$
  endif
endfor

Output: A partitioning on the noun phrases NP_1, NP_2, ..., NP_k.

Figure 3.3: The Closest-First Clustering Algorithm

aliasing), and one positional feature (number of sentences between the two NPs). Unlike McCarthy and Lehnert, the *non-transitive* method is used to generate positive training instances from coreference chains. To reduce the ratio of negative instances to positive instances (and hence the skewness of the class distributions), only a subset of the available negative instances is used for training: a negative instance is created by pairing an anaphoric NP, NP_j, with each NP appearing between NP_j and its closest preceding antecedent in the associated text. A left-to-right single-link clustering algorithm that selects the closest preceding coreferent NP as the antecedent for each anaphoric noun phrase is employed to create coreference equivalence classes (see Figure 3.3). Results indicate that STR_MATCH (i.e., whether the surface strings of the two NPs are the same after determiners are

discarded), ALIAS (i.e., whether two named entities are aliases)[5], and APPOSITIVE (i.e., whether the two NPs form an appositive construction) are strong indicators of coreference. The system is applied to two standard coreference data sets (MUC-6, 1995; MUC-7, 1998), achieving performance comparable to the best-performing knowledge-based coreference engines.

Strube et al.'s anaphora resolution system for narrative texts and spoken dialogues. Strube et al. (2002) describe a learning approach to resolution of German anaphors using a decision tree induction system. Their work was motivated by the observation that the feature set commonly employed in existing anaphora/coreference systems tends to produce low recall (i.e., many anaphora/coreference relationships are missed by these systems). In particular, they notice that many of these relationships are formed between NPs whose surface strings resemble but are not identical to each other, adding that the string matching facilities in existing systems are not sophisticated enough to discover these relationships. As a result, they propose two easily computable lexical features that measure the *minimum edit distance* between the two NPs under consideration, with the goal of recovering the anaphoric/coreference relationships between lexically similar strings. (See Section 4.5 for a detailed description of how these two features are computed.)

In a more recent paper (Strube and Müller, 2003), they extend their system to handle the resolution of pronominal NPs in spoken dialogues. Roughly speaking, there are two major extensions. First, the system attempts to resolve not only

[5]Here, different types of aliasing are defined for different named-entity types: two dates are aliases if their normalized form is identical; two persons are aliases if the last word of the two NPs involved are identical; and two organizations are aliases if one is an acronym of the other.

pronouns with NP antecedents but also those with non-NP antecedents, a capability that is absent in virtually all existing pronoun resolution systems. Second, the system incorporates an additional set of features that is tailored to resolving pronouns in spoken dialogues. In particular, some of these features aim to distinguish NP from non-NP antecedents, whereas others encode the kind of complement that the verb governing the pronoun under consideration tends to subcategorize for.

Kehler's work on probabilistic coreference. Rather than performing coreference resolution before generating templates in a standard IE architecture (Cardie, 1997), Kehler's coreference system (Kehler, 1997b) operates on the output of an IE system in which one template is produced for each entity mentioned in the document. The goal of Kehler's system is then to generate a probability distribution over all possible partitions of the output templates such that the templates in each equivalence class are coreferent. The coreference system derives a maximum entropy model (Berger et al., 1996) for determining the probability that two templates co-refer based on three types of features. The first type of feature tests whether two templates possess a subsumption relationship by checking whether the relationship is satisfied for each slot in the template. The second type of feature is based on the definiteness of the entity associated with one of the templates. The third type of feature measures the distance between the entities associated with the templates. Training instances are generated from all pairs of templates. The system then uses *Dempster's Rule of Combination* (Dempster, 1968) to combine the pairwise coreference probabilities generated from the learned model to score a partition. The resulting scores (with one score per partition) are normalized to form a probability distribution over the set of possible partitions. Because the

number of possible partitions grows exponentially with the number of templates[6], it is unclear whether Kehler's approach is scalable to a large number of templates.

Connolly et al.'s preference-trained decision trees. In all of the learning-based coreference systems described above, coreference is recast as a classification task, where each training instance represents two NPs and is assigned a label of COREFERENT or NOT COREFERENT depending on whether the two NPs are co-referring or not in the associated text. Connolly et al. (1994) argue that it is better to formulate the problem as one that determines the best antecedent for a given anaphor, and unlike previous work, they present an approach that explicitly captures the relative preference between two potential candidates for a given anaphor. Specifically, each training instance represents an anaphor as well as two of its candidate antecedents (one is correct and the other is not). In this case, the label is a binary value indicating which of the two candidates is the correct antecedent. Yang et al. (2003) and Iida et al. (2003) adopt essentially the same approach to training a preference-based classifier for coreference resolution and pronoun resolution, respectively. Noting that the number of training instances formed from a given anaphor is quadratic in the number of candidate antecedents, Yang et al. (2003) introduce heuristics for selecting training instances.

Antecedent selection (during testing) is performed differently in these three

[6]According to Anderberg (1973), the number of ways to partition n items is

$$\sum_{k=1}^{n} S_n^{(k)}$$

where

$$S_n^{(k)} = \frac{1}{k!} \sum_{i=0}^{k} (-1)^{k-i} \binom{k}{i} i^n.$$

systems. In Connolly et al. (1994) and Iida et al. (2003), candidate antecedents for a given anaphor are considered in a right-to-left manner as follows. Initially, the learned classifier is used to determine which of the two rightmost candidates is preferred. The "winner" is then compared with the rightmost unconsidered candidate, and the process is repeated between the winner from the most recent comparison and the rightmost unconsidered candidate. On the other hand, Yang et al.'s (2003) antecedent selection algorithm maintains for each candidate antecedent a score that indicates the number of times a given candidate is preferred over the other candidates, selecting the one that has the highest score as the antecedent.

Finally, with respect to the learning algorithm employed, Connolly et al. (1994) and Yang et al. (2003) use a decision tree induction system, whereas Iida et al. (2003) rely on support vector machines (Cristianini and Shawe-Taylor, 2000).

3.3.3 Statistical Approaches

As a subclass of corpus-based approaches, statistical approaches to anaphora resolution attempt to construct a conditional model that determines the probability that an NP is an antecedent of a given anaphor in the presence of a set of potentially relevant factors. To improve the robustness of the statistics collected from a given corpus, the set of conditioning factors is decomposed into smaller "pieces" by using Bayes rule and making certain independence assumptions.

Ge et al.'s statistical approach. Ge et al. (1998) present a probability model for resolving third-person pronouns. Four training features for the model are used, including positional information (i.e., the distance between the pronoun and the candidate antecedent), grammatical information (i.e., gender and animacy of the

candidate antecedent), semantic information (i.e., selectional preferences based on the governing constituent of the pronoun), and a crude measure of salience (i.e., mention count of the candidate antecedent). Ge et al. show how the equation for the model can be decomposed into these four factors for which statistics can be collected directly from the training corpus. For a given anaphoric pronoun, the candidate antecedent that is assigned the highest probability by the model is selected as the antecedent of the pronoun. Their results demonstrate that knowledge of the candidate antecedent's gender and animacy and its salience are essential to anaphora resolution.

So far we have provided an overview of previous work on anaphora and coreference resolution and characterized each system as either knowledge-based or corpus-based. Instead of performing a similar characterization on a system basis along the remaining dimensions, we will point out some of the key issues involved in each of these dimensions and focus on the strengths and weaknesses of the related approaches. In Section 3.4, we will compare knowledge-rich approaches with knowledge-lean approaches. In Section 3.5, we will look at the implications of semi-automatic pre-processing and fully automatic pre-processing. Section 3.6 discusses the issues with respect to small-scale evaluation versus large-scale evaluation of anaphora and coreference resolution systems.

3.4 Knowledge-Rich versus Knowledge-Lean Approaches

As we mentioned above, anaphora and coreference resolution systems can in principle be ordered according to the *types* of knowledge source required to perform accurate resolution: knowledge-rich or knowledge-lean. Early anaphora resolution

systems such as Grosz (1977), Sidner (1979), and Carbonell and Brown (1988) are knowledge-rich systems that rely on domain information as well as semantic and pragmatic knowledge that is often hard to represent and process and needs to be encoded for each new domain. Knowledge-lean approaches, which only makes use of morphological and shallow syntactic information, are developed precisely to meet the need for efficient solutions to anaphora resolution. In particular, knowledge-lean anaphora resolution systems such as Lappin and Leass (1994), Kennedy and Boguraev (1996), Baldwin (1997) and Mitkov (1998) have all demonstrated that high-performance anaphora resolution is possible without semantic and world knowledge. Although Lappin and Leass's algorithm assumes the output of a full parser, Kennedy and Boguraev show that the system only suffers from a small drop in accuracy when the parser output is approximated using a set of heuristics.

Given the successful shift from knowledge-rich approaches to knowledge-lean approaches for the task of anaphora resolution in recent years, two immediate questions arise:

- How much extra benefit can be obtained if the system has access to semantic information? Equivalently, how much performance loss is incurred by relying only on morphological and shallow syntactic information?

- Does the same result generalize to coreference resolution? In other words, is high-performance knowledge-poor coreference resolution possible?

The work by Ge et al. (1998) sheds some light on the first question. Specifically, one of the factors that their statistical pronoun resolution system depends on is selectional restrictions, which is a form of semantic knowledge. They evaluate

their system by incorporating the knowledge sources incrementally into the system. Their results indicate that the system that has access to semantic knowledge only performs marginally better than the one without semantic knowledge. However, they also remark that selectional restrictions "are not clearcut in most cases" and that "some verbs are too general to restrict the selection of any NP".

On the other hand, the answer to the second question seems to be different under different circumstances. For domain-specific coreference resolution, both Aone and Bennett (1995) and McCarthy and Lehnert (1995) report that the decision tree classifiers take advantage of domain-specific semantic information. In contrast, the results of Soon et al. (2001) seem to imply that a general-purpose coreference resolution can achieve good performance without semantic information. Specifically, their decision tree classifier relies almost entirely on string matching, apposition, and aliasing for recognizing coreference relationships, despite the fact that it has access to information regarding semantic class compatibility. Although aliasing is generally viewed as a semantic knowledge source, the corresponding feature value is actually computed via a set of simple lexical matching routines without resorting to any semantic knowledge base. Nevertheless, the fact that the semantic feature is not selected by Soon's decision tree is by no means an indication that semantic information is irrelevant to coreference resolution, since the error rate resulting from the automatic computation of the feature has not been measured.

Despite the popularity of knowledge-lean approaches, many of the MUC coreference systems make use of semantic information provided by WordNet (Fellbaum, 1998). In general, the availability of general-purpose lexical resources such as WordNet and advances in research in named-entity classification have made it convenient for anaphora and coreference resolution systems to incorporate semantic

information. It should be noted, however, that systems using semantic information from WordNet need to somehow deal with the word sense disambiguation problem, which is widely known to be a hard problem in NLP. Consequently, the issue regarding the accuracy of semantic feature computation remains.

3.5 Semi-Automatic versus Fully Automatic Pre-Processing

Mitkov (2001) points out that research in anaphora resolution has "suffered from a bizarre anomaly in that until recently hardly any fully automatic operational systems had been reported". For example, Lappin and Leass (1994) post-edit the outputs of the part-of-speech tagger and the full parser on which their anaphora resolution algorithm relies. Similarly, Mitkov (1998) manually fixes the outputs of the part-of-speech tagger and the base noun phrase finder. In addition, the knowledge assumed by the resolution algorithms described in Mitkov (1997) such as gender is obtained directly from the data sets that are annotated with the corresponding information. Some exceptions exist, however. The BFP anaphora resolution algorithm (Brennan et al., 1987) is fully automatic, for example; the gender and number information required by their centering algorithm is provided by other modules in their HPSG natural language system.

The MUC-6 and MUC-7 conferences (MUC-6, 1995; MUC-7, 1998) have promoted the development of coreference resolution systems that operate in fully automatic mode. In addition, virtually all coreference systems developed since MUC-6 (e.g., Aone and Bennett (1995), McCarthy and Lehnert (1995)[7], Soon

[7]The feature values provided to this coreference system are all computed manually. The authors claim that this makes it possible to evaluate the errors made

et al. (2001)) as well as the CogNIAC pronoun resolution system (Baldwin, 1997), which was part of the CAMP system participating in the MUC coreference task (Baldwin et al., 1998), are also fully automatic. Furthermore, the emergence of statistical approaches has obviated the need for certain kinds of preprocessing. For example, Ge et al. (1998) describe a method for automatically computing gender information from an annotated corpus.

The reason for the shift from semi-automatic pre-processing to fully automatic-preprocessing is that the former is practically unrealistic. Specifically, it is impractical to ask a human to stand by and fix all errors made by pre-processing routines. Even if the pre-processing taggers are all highly reliable and only a small number of errors needs to be fixed manually, the human still has to go through all the outputs of the taggers in order to locate the errors. This proves to be a time-consuming task, especially if the system relies on a large number of knowledge sources or needs to process a large number of documents.

On the other hand, feeding an anaphora or coreference algorithm with perfectly computed knowledge has the advantage that the performance of the resolution algorithm is not affected by errors propagated from upstream pre-processing components. Consequently, we can easily measure the errors made by the resolution algorithm versus the errors made by the pre-processing routines.

Recent work has shown that the performance of resolution algorithms is severely limited by the accuracy with which the knowledge required by the algorithms can be computed, thus providing suggestive evidence that the task has been made much easier in the presence of semi-automatic pre-processing. In particular, a recent re-implementation of Mitkov's robust pronoun resolution algorithm (Mitkov, 1998)

---

by the coreference resolution *algorithm* and that errors made by pre-processing routines should be evaluated separately.

and evaluation on new data sets by Mitkov et al. (2002) shows that the performance of the new, fully automatic algorithm is not as high as the initial results obtained using the semi-automatic algorithm. As a result, it makes sense to take into account the *robustness* of a resolution algorithm to potentially erroneous input when measuring its performance. As a side note, there are two classes of algorithms that are in principle more robust than the others:

- Knowledge-lean algorithms: As noted in Section 1.2, an anaphora or coreference resolution algorithm can potentially rely on a large number of linguistic knowledge sources. With respect to the accuracy of knowledge sources, lower-level knowledge such as lexical and syntactic knowledge can be computed with comparatively better accuracies than higher-level knowledge such as semantic and pragmatic knowledge. In fact, except possibly for lexical string matching which can be performed without any error, none of the knowledge sources can be computed with perfect accuracy. Consequently, knowledge-lean resolution algorithms that usually depend on just the outputs of a part-of-speech tagger and a base noun phrase finder (e.g., Baldwin (1997), Mitkov (1998)) tend to be more robust than their knowledge-rich counterparts, especially since the corresponding tasks can be accomplished with fairly high accuracies.

- Corpus-based algorithms: Corpus-based algorithms are trained on real-world texts and hence are by nature more robust in the face of incomplete and noisy input than their knowledge-based counterparts. For example, we mentioned in Section 1.2 that although there are exceptions to traditional linguistic constraints such as number agreement, many knowledge-based systems such as Lappin and Leass (1994) and Kennedy and Boguraev (1996) simply encode

them as broadly and unconditionally hard constraints and thus effectively ignore the exceptions. In contrast, corpus-based algorithms learn soft constraints empirically from labeled data and can therefore take into account these exceptions.

3.6 Small-Scale versus Large-Scale Evaluation

For most early anaphora resolution algorithms, evaluation is done by hand on a carefully chosen small set of documents, due in part to the fact that these algorithms were not implemented as computer programs at the time of evaluation. For example, Hobbs evaluates his algorithm (Hobbs, 1978) on 300 pronouns extracted from three different sources. The comparative evaluation by Walker (1989) on the Hobbs algorithm and the BFP algorithm is performed on 281 third-person pronouns in articles from three different genres.

One advantage of manual evaluation on small data sets over automatic evaluation is that it offers a better picture of what mistakes are made by an algorithm and how the algorithm can be improved. On the other hand, evaluation on small data sets means that the resulting performance of the algorithm may not accurately reflect its actual performance, at least from a statistical perspective. The recent explosion of the number of online texts has made large-scale, automatic evaluation possible.

3.7 Chapter Summary

In this chapter, we have given an overview of related work on anaphora and coreference resolution in the context of the generic algorithm described in the previous

chapter. We start by describing four major research trends in anaphora and coreference resolution: from knowledge-rich approaches to knowledge-lean approaches; from semi-automatic pre-processing to fully automatic pre-processing; from small-scale evaluation to large-scale evaluation; and from knowledge-based approaches to corpus-based approaches. We then present a detailed characterization of related work along the last dimension.

In the next chapter, we will present our learning-based coreference resolution system, with the goal of improving existing machine learning approaches to coreference resolution. We will compare and contrast the design of our system with that of the learning-based coreference systems discussed in Section 3.3.2, discussing why our system can potentially yield better performance than existing approaches to coreference resolution.

CHAPTER 4

OUR COREFERENCE RESOLUTION SYSTEM

> *It can be said that the corpus-based study of discourse anaphora is in its infancy. Other corpus-based computational areas of research, such as parsing, and morphosyntactic analysis, are well-developed, but anaphora studies using corpora are still working on first principles, and there is no unified approach as yet.* — Botley and McEnery (2000)

Recall that our ultimate goal is to improve existing machine learning approaches to coreference resolution. To this end, we will propose a set of extensions to the standard machine learning framework consisting of classification and clustering. As we will see, all of our proposed extensions are motivated by potential problems with the existing learning-based coreference systems discussed in Section 3.3.2. Perhaps more importantly, none of these extensions is specific to a particular coreference system implementation. In other words, they are all applicable to any coreference system implementing the standard machine learning framework.

Roughly speaking, the proposed extensions can be divided into two types: **linguistic extensions** and **extra-linguistic extensions**. The linguistic extensions take into account what linguistic theories can offer regarding discourse anaphora and coreference. The extra-linguistic extensions, on the other hand, aim at improving the standard machine learning framework and are motivated primarily from a machine learning perspective.

In the rest of this chapter, we will examine each of these extensions and discuss how to integrate them into our coreference system.[1] Specifically, Sections

[1]Portions of this chapter have appeared elsewhere. Section 4.3 and portions of Section 4.2 are based on Ng and Cardie (2002a). Section 4.5 is adapted from Ng

4.1-4.4 propose four extra-linguistic extensions involving the *clustering algorithm*, the method of *training instance creation*, the *loss function* employed in optimizing the coreference classifier, and the learning algorithm. Sections 4.5-4.6 consider two linguistic extensions to the standard machine learning framework for coreference resolution. The first linguistic extension involves a large-scale expansion to the knowledge sources commonly employed by learning-based coreference systems. The second one involves computing and using anaphoricity information to improve coreference resolution. For each of the above extensions, we will begin with an overview of the relevant issues, which, as mentioned before, are all motivated by potential problems with existing learning-based coreference systems. As a result, we will revisit the design of specific components in these systems as needed. Then, in Section 4.7, we propose a simple algorithm for integrating all of our proposed extensions into our learning-based coreference system. Finally, Section 4.8 summarizes our discussion in this chapter.

4.1 Clustering Algorithm

Our first extra-linguistic extension involves the clustering algorithm. Recall that the clustering mechanism is used to coordinate the possibly contradictory pairwise classification decisions made by a coreference classifier and construct a partition on a given set of NPs. In particular, since coreference is typically viewed as a problem of selecting an antecedent(s) for each potentially anaphoric NP, clustering for coreference resolution is essentially an antecedent selection procedure.

As seen in Section 3.3.2, a variety of clustering algorithms have been employed

and Cardie (2002c). Sections 4.6.1 and 4.6.2 are based on Ng and Cardie (2002b) and Ng (2004), respectively.

by existing learning-based coreference systems. For instance, Soon et al. (2001) use a **closest-first** clustering algorithm (see Figure 3.3), which selects as the antecedent for an NP the closest preceding noun phrase that is classified as coreferent with it. On the other hand, Aone and Bennett (1995) rely on a **best-first** clustering algorithm (see Figure 3.1) to select as the antecedent for an NP the "most likely" preceding coreferent noun phrase, using the confidence value associated with the classifier's prediction as a measure of likelihood. Finally, McCarthy and Lehnert's (1995) **aggressive-merge** clustering algorithm (see Figure 3.2) merges each NP with all of its preceding noun phrases that are classified as coreferent with it.

It is conceivable that these three clustering algorithms would give rise to different levels of coreference performance. Aone and Bennett's best-first clustering algorithm, for instance, can be expected to produce partitions with higher precision than Soon et al.'s, since, given a set of candidate antecedents, the most confident one is more likely to be the correct choice than the closest one. On the other hand, the clustering algorithm employed by McCarthy and Lehnert's system merges the clusters more aggressively than the other two, and therefore should generate partitions with higher recall.

A natural question to ask is which clustering algorithm is the best one to use. As seen above, existing learning-based coreference systems often adhere to a particular clustering algorithm without justifying their choice. However, such ad-hoc design decisions may result in sub-optimal system performance — a potential problem with existing approaches.

Given this observation, we propose to consider multiple clustering algorithms in our coreference system, and adopt a corpus-based approach to select the "best"

clustering algorithm among the three discussed in this section. We will defer the discussion of this approach to Section 4.7.

Finally, and perhaps most importantly, the fact that we consider only three clustering algorithms in our experiments is by no means a self-imposed limitation of our approach: our framework allows the incorporation of as many clustering algorithms as we desire. The above three clustering algorithms are chosen primarily because they are commonly employed in existing learning-based coreference systems. There are certainly other clustering techniques that one may want to consider applying to the coreference task. In **correlation clustering** (Bansal et al., 2002), for instance, the clustering algorithm is given a set of objects to be clustered as well as a similarity metric that classifies two objects as either IN THE SAME CLUSTER or NOT IN THE SAME CLUSTER. The goal then is to generate a partition on the objects that respects as many pairwise classification decisions as possible. This distinguishes correlation clustering from our three clustering algorithms, which make no attempt to minimize the perturbation to the original classification decisions during the partitioning process.

4.2 Training Instance Creation

Our second extra-linguistic extension involves training instance selection. Recall that to employ the supervised learning paradigm to automatically acquire a classifier, the learning algorithm assumes as input a set of training instances. As seen in Section 3.3.2, different methods for creating coreference training instances have been proposed by existing machine learning approaches to the problem. McCarthy and Lehnert's method, repeated below for the sake of convenience, is perhaps the simplest one.

McCarthy and Lehnert's training instance creation method. A *positive instance* is created for each anaphoric NP paired with each of its antecedents; and a *negative instance* is created by pairing each NP with each of its preceding non-coreferent noun phrases.

Note that, in this method, the number of positive instances created from an anaphoric noun phrase, NP_j, is equal to the number of preceding NPs that are coreferent with NP_j. However, as noted in Section 3.3.1, one antecedent is sufficient for resolving an anaphor. In other words, it may be sufficient to create one positive instance from each anaphoric noun phrase. This motivates the Aone and Bennett training instance creation method, as shown below.

Aone and Bennett's training instance creation method. A *positive instance* is created for each anaphoric NP and its closest antecedent; and a *negative instance* is created by pairing each NP with each of its preceding non-coreferent noun phrase.

There is a potential problem with these two methods for creating training instances: the resulting coreference data set would be highly imbalanced. The reason is that coreference is a *rare* relation, which means that most of the NP pairs in a document are *not* coreferent. Consequently, generating training instances from all NP pairs as in McCarthy and Lehnert creates highly skewed class distributions, in which the negative instances significantly outnumber the positive instances. Aone and Bennett's method further aggravates the problem by creating an even smaller number of positive instances than that of McCarthy and Lehnert. Unfortunately, learning from highly imbalanced data sets remains an open area of research in the machine learning community (e.g., Pazzani et al. (1994), Fawcett (1996), Cardie

and Howe (1997), Kubat and Matwin (1997)).

Most of the existing methods for handling skewed class distributions modify the learning algorithm to incorporate a loss function with a much larger penalty for minority class errors than for instances from the majority classes (e.g., Gordon and Perlis (1989), Pazzani et al. (1994)). Nevertheless, Soon et al. adopt a different approach to handling skewed class distributions — negative instance selection, i.e., the selection of a smaller subset of negative instances from the set of available negative instances. As shown below, Soon et al.'s method differs from that of Aone and Bennett precisely in its reduction in the number of negative instances produced.

Soon et al.'s training instance creation method. A *positive instance* is created for each anaphoric NP, NP_j, and its closest antecedent, NP_i; and a *negative instance* is created for NP_j paired with each of the intervening NPs, NP_{i+1}, NP_{i+2}, ..., NP_{j-1}.

Despite its ability to alleviate the problem of data skewness, Soon et al.'s method creates a much smaller set of (positive and negative) training instances in comparison to that of McCarthy and Lehnert as well as Aone and Bennett. Consequently, it is possible that too much potentially useful information is lost during the instance creation process.

As a result, we propose a parametric family of training instance creation schemes that lies between the two extremes exemplified by McCarthy and Lehnert's method and Aone and Bennett's method, respectively. Specifically, our methods attempt to strike a balance between reducing data skewness and minimizing information loss by progressively creating additional training instances on top of the Soon et al.

method. As shown below, our Soon-n instance creation scheme generalizes Soon's method via the introduction of the parameter n, yielding Soon precisely when n is equal to 1.

Our Soon-n training instance creation methods. A *positive instance* is created between an anaphoric NP, NP_j, and each of its n closest antecedents. Now, assuming that NP_i is the n-th closest antecedent of NP_j, a *negative instance* is created for NP_j paired with each of the intervening non-coreferent NPs, NP_{i+1}, NP_{i+2}, ..., NP_{j-1}.

Assuming that the amount of information contained in a data set correlates positively with its size, we can see that the value of n represents a trade-off between reducing data skewness and minimizing information loss: as n increases, more information is preserved, but data skewness also becomes more serious due to the faster rate at which the negative instances increase when compared to the positive instances.

Given the Soon-n training instance creation methods as well as the ones previously proposed by McCarthy and Lehnert, Aone and Bennett, and Soon et al., a natural question to ask is which of these methods is the best one to use. Like the selection of the "best" clustering algorithm, we will propose a corpus-based solution to this question in Section 4.7. Automatically selecting the "best" instance creation method has the advantage of avoiding the ad-hoc, possibly sub-optimal decisions made in existing learning-based approaches, which often adhere to an instance creation scheme without justifying the particular choice.

Other training instance creation methods. Like the clustering algorithms, the fact that we consider only the above training instance creation methods does not reflect a self-imposed limitation of our approach. Of course, one can possibly conceive of other training instance creation methods. To further alleviate the problem of skewed class distributions, for instance, we may have a simple variant of the Soon-n methods in which the positive instances are created as in Soon-n, but the negative instances are created according to Soon-1. It should be fairly easy to see that this Soon-n variant produces class distributions that are less skewed than the corresponding Soon-n method for all $n > 1$.

The existing training instance selection methods for coreference that we decided not to consider in our experiments include **NegSelect** and **PosSelect**, which are previously proposed by us for performing negative instance selection and positive instance selection, respectively (Ng and Cardie, 2002a).[2] **NegSelect** is shown in Figure 4.1. Given the set of all possible negative instances in the training set (i.e., the set of instances $inst_{(NP_i,NP_j)}$ such that NP_i and NP_j are not in the same coreference chain), **NegSelect** only retains only those negative instances for non-coreferent NPs that lie between NP_j and its *farthest* antecedent.

PosSelect, shown in Figure 4.2, is motivated by the observations that (1) one antecedent is sufficient for resolving an anaphor and (2) not all of the coreference relationships are equally easy to identify (e.g., it is generally harder to identify pronominal antecedents than non-pronominal antecedents for definite de-

[2]The two terms *instance creation* and *instance selection* both refer to the process of forming a training set for learning a classifier. When we say *instance creation*, we are thinking of a bottom-up process in which we create a set of training instances from scratch. When we say *instance selection*, we are thinking of a top-down process in which we select a smaller subset of the instances from the set of available training instances.

NegSelect(NEG)

Input: A set of all possible negative instances NEG

Algorithm:
foreach $inst_{(NP_i,NP_j)} \in NEG$ **do**
  if NP_j is anaphoric **then**
    if NP_i precedes the farthest antecedent of NP_j **then**
      $NEG := NEG \setminus \{inst_{(NP_i,NP_j)}\}$
  else
    $NEG := NEG \setminus \{inst_{(NP_i,NP_j)}\}$

Output: NEG

Figure 4.1: The **NegSelect** Algorithm

scriptions). Hence, it may be desirable to learn a coreference classifier on a subset of the available positive training instances. In fact, as mentioned in Section 3.3.1, Harabagiu et al. (2001) point out that intelligent selection of positive instances can potentially minimize the amount of knowledge required to perform accurate coreference resolution. They assume that the easiest types of coreference relationships to resolve are those that occur with high frequencies in the data. Consequently, they mine by hand three sets of coreference rules for covering positive instances from the training data by finding the coreference knowledge satisfied by the largest number of antecedent-anaphor pairs. **PosSelect** attempts to automate this positive instance selection process. More precisely, it coarsely mimics the Harabagiu et al. algorithm by finding a *confident* antecedent for each anaphor. As shown in Figure 4.2, **PosSelect** assumes the existence of a rule learner, L, that produces an ordered set of *positive* rules. The algorithm first uses L to induce a ruleset on the training instances T and picks the first rule from the ruleset. For any training

---

PosSelect(L, T)

Input: A positive rule learner L, and a set of training instances T

Algorithm:
$FinalRuleSet := \emptyset$
$AnaphorSet := \emptyset$
$BestRule :=$ NIL

repeat
  $BestRule :=$ best rule among the ranked set of rules induced on T using L
  $FinalRuleSet := FinalRuleSet \cup BestRule$

  ◁ Collect anaphors from instances that are correctly covered by $BestRule$ ▷
  foreach $inst_{(NP_i,NP_j)} \in T$ **do**
    if $inst_{(NP_i,NP_j)}$ is covered by $BestRule$ **and**
    class($inst_{(NP_i,NP_j)}$) = COREFERENT **then**
      $AnaphorSet := AnaphorSet \cup \{$ NP$_j\}$

  ◁ Remove instances associated with the anaphors covered by $BestRule$ ▷
  foreach $inst_{(NP_i,NP_j)} \in T$ **do**
    if NP$_j \in AnaphorSet$ **then**
      $T := T \setminus \{inst_{(NP_i,NP_j)}\}$

until L cannot induce any rule for the positives

Output: $FinalRuleSet$

Figure 4.2: The **PosSelect** Algorithm

instance $inst_{(NP_i,NP_j)}$ correctly covered by this rule, an antecedent NP_i has been identified for the anaphor NP_j. As a result, all (positive and negative) training instances formed with NP_j as the anaphor are no longer needed and are subsequently removed from the training data. The process is repeated until L cannot induce a rule to cover the remaining positive instances. The output of **PosSelect** is a set of positive rules selected during each iteration of the algorithm. Hence, positive sample selection in this algorithm is implicit in the sense that it is embedded within the rule induction process.

Previous results on the MUC-6 and MUC-7 coreference data sets indicate that these two methods, when used in combination with the rule-pruning algorithm to be discussed in the next section, are effective in producing a high-performing coreference system. We do not consider these two algorithms in our current work simply because **PosSelect** is not as scalable as we want: the algorithm becomes fairly inefficient when applied to some of our large evaluation data sets.

4.3 Rule Pruning

Our third extra-linguistic extension involves the optimization of the automatically acquired coreference classifier. Machine learning algorithms typically train classifiers to maximize classification accuracy on unseen data. However, this is presumably a problem for coreference resolution, where our ultimate goal is to maximize a coreference classifier's accuracy at the clustering level. More specifically, the problem is that there is no direct correlation between maximizing classification accuracy and maximizing clustering accuracy. In other words, improvements in classification accuracy do not necessarily guarantee corresponding improvements in clustering accuracy. Unfortunately, existing machine learning approaches to the

problem make no attempt to address this potential problem.

In this section, we propose a method for resolving this conflict. Specifically, given a coreference classifier that can be expressed in the form of a ruleset, we use an *error-driven rule-pruning algorithm* to discard those rules that cause the ruleset to perform poorly with respect to the global, clustering-level coreference scoring function.

Error-driven rule pruning. The goal behind rule pruning is to find a subset of a given set of rules that would perform well on unseen data. As mentioned above, our rule-pruning algorithm is error-driven, meaning that the goal is to discard rules whose removal yields performance gains on unseen data according to a given loss function. In principle, any (labeled) unseen data that is disjoint from the test data can be used as our *pruning* corpus. However, as we will see below, we may still be able to obtain additional benefits by using "seen" data for pruning rules induced by a coreference classifier.

The rule-pruning algorithm. The error-driven pruning algorithm, **RuleSelect**, is shown in Figure 4.3. It takes as input a ruleset learned from a training corpus for performing coreference resolution, a pruning corpus, and a clustering-level coreference scoring function that is the same as the one being used for evaluating the final output of the system. At each iteration, **RuleSelect** greedily discards the rule whose removal yields a ruleset with which the coreference system performs the best (with respect to the coreference scoring function) on the pruning corpus. As a hill-climbing procedure, the algorithm terminates when removal of any of the rules in the ruleset fails to improve performance. In contrast to most existing algorithms for coreference resolution, **RuleSelect** establishes a tighter connection

RuleSelect(R, P, S)

Input: A ruleset R, a pruning corpus P, and a scoring function S

Algorithm:
$BestScore$:= score of the coreference system using R on P with respect to S;
r := NIL;

repeat
  r := the rule in R whose removal yields a ruleset with which the coreference system achieves the best score b on P with respect to S.
  if $b > BestScore$ **then**
    $BestScore := b$;
    $R := R \setminus \{r\}$
  else
    break
while true

Output: R.

Figure 4.3: The **RuleSelect** Algorithm

between the classification- and clustering-level decisions for coreference resolution and ensures that system performance is optimized with respect to the coreference scoring function.

Now, let us make a few observations about our rule pruning algorithm.

- While we expect that rule pruning will be most effective if the pruning corpus is disjoint from the training corpus, we may still be able to obtain benefits even by reusing the training corpus for pruning. This is again due to the fact that the loss function employed by the clustering-level scoring function is different from the one used by the learning algorithm. In other words, the set of rules that achieves the highest classification accuracy on a given set

of labeled data is not necessarily the same as the set of rules that achieves the highest clustering-level coreference accuracy on the same data. This flexibility of (partially or entirely) reusing the training corpus for pruning is particularly useful when labeled data is scarce, in which case we might not be able to afford to set aside our labeled data for pruning purposes.

- **RuleSelect** assumes *no* knowledge of the inner workings of the scoring function. This implies that the algorithm can in principle be used to optimize *any* coreference scoring function.

- **RuleSelect** assumes *no* knowledge of how the input coreference ruleset is induced. This broadens the applicability of the algorithm to rulesets acquired by both knowledge-based and learning-based approaches to coreference resolution.

- **RuleSelect** bears resemblance to the backward elimination algorithm commonly used for feature selection (see Kohavi and John (1997)) in terms of both the ultimate goal and the algorithmic design. In both cases, the goal is to select a subset of a given set of objects (which are rules in the case of rule pruning and features in the case of feature selection). To achieve this goal, both algorithms adopt a greedy approach, iteratively discarding the object whose removal yields the best performance on the task at hand.

4.4 Learning Algorithm

In Sections 4.1 and 4.2, we mentioned that existing learning-based coreference systems make ad-hoc decisions by adhering to a specific clustering algorithm and training selection method without considering other alternatives. The same po-

tential problem applies to their choice of the underlying learning algorithm for training a coreference classifier.[3] Following the line of thought we have established thus far, we decide to consider two different learners in our coreference system — our fourth extra-linguistic extension — as described below.

4.4.1 RIPPER

The first learner we consider is a rule-based learner. Recall that the rule-pruning algorithm described in the previous section is only applicable to classifiers that can be expressed in the form of a ruleset. Thus, it is desirable that we choose a rule-learning algorithm for acquiring a coreference classifier.

Specifically, we employ RIPPER (Cohen, 1995) for acquiring a set of *propositional* rules, each of which is comprised of a *condition* expressed as a conjunction of attribute-value pairs and a *classification value*. Specifically, during training, RIPPER induces a set of (positive) rules that determines when two NPs are coreferent, where each rule is associated with a *confidence value* that reflects the accuracy of the rule on the training data. Then, during testing, a test instance considers each of the induced rules in turn. If the instance satisfies the conditions specified by a rule (meaning that each attribute-value pair appearing in the rule also appears in the instance), then the classifier returns a classification value of COREFERENT. On the other hand, if none of the induced rules is applicable to a given NP pair, a

[3]Nevertheless, although existing work appears to have a diverse opinion on which clustering algorithm and training creation method to use, this is not the case for learning algorithms. With the exception of Kehler (1997b) and Iida et al. (2003), who make use of maximum entropy modeling and support vector machines respectively, decision tree learners are the learning algorithm of choice for essentially all of the existing learning-based coreference engines (e.g., Aone and Bennett (1995), McCarthy and Lehnert (1995), Soon et al. (2001), Strube et al. (2002), Yang et al. (2003)).

default rule that classifies the pair as NOT COREFERENT is automatically invoked. In both of the above cases, the confidence associated with the rule is returned in conjunction with the classification value.

4.4.2 C4.5

The second learner we consider is C4.5 (Quinlan, 1993), a decision tree learner. As mentioned above, decision tree learners have been employed extensively and reasonably successfully in existing learning-based coreference engines. This motivates us to consider C4.5 as one of the learners for training our coreference classifiers.

The goal of decision tree learning is to recursively partition the instance space until (most of) the training instances in the same cluster of the partition are of the same class. This is achieved by having the the decision tree learner greedily add the most informative feature to an initially empty tree structure. More specifically, note that each feature f defines a partition P_f on the instance space, where each cluster in P_f contains exactly the instances that share the same feature value with respect to f. Hence, once we select a feature f for addition to the tree structure, we can repeat this process to recursively partition each cluster defined by f.

By virtue of this recursive partitioning process, the decision tree learner generates a tree structure. Each non-leaf node in the tree corresponds to a feature, representing a choice point among the possible values of the feature. A choice can then be made by traversing one of the outgoing edges of the node. Hence, underlying each path from the root to a leaf l of the tree is a conjunction of feature-value pairs satisfied by all of the instances in l. Moreover, each l is associated with a *class label* c_l (i.e., the majority class of the instances in l) and a *confidence value* that specifies the accuracy of classifying all instances in l as c_l. During testing,

an unseen instance starts from the root of the tree and recursively traverses the appropriate outgoing edge until it reaches a leaf node. The instance then receives the class label associated with the leaf node. All implementations of decision tree learners, including C4.5, are variations of this basic algorithm.

As mentioned above, each path from the root to a leaf of a decision tree corresponds to a conjunction of attribute-value pairs, and each leaf is associated with a class label. Hence, a tree can be equivalently represented as a ruleset with one rule for each such path-label combination. This implies that the rule-pruning algorithm can be applied to a decision tree classifier. Implementation-wise, however, it is tedious to convert a decision tree into an equivalent ruleset. As a result, we will only apply rule pruning to classifiers acquired by RIPPER for the sake of simplicity. Evaluation details will be given in the next chapter.

4.5 Feature Set Enhancement

So far we have considered four extra-linguistic extensions to the standard machine learning framework for coreference resolution. In this section and the next, we will consider two *linguistic* extensions.

Our first linguistic extension involves a large-scale expansion of the features available to the learning algorithm. Most existing learning-based coreference systems rely on a fairly small set of features for training a coreference classifier: McCarthy and Lehnert (1995), Soon et al. (2001), and Müller et al. (2002) use 8, 12, and 16 features in their feature-vector representation, respectively.[4] Given that coreference is a complex linguistic phenomenon, a natural question to ask

[4]One exception is Aone and Bennett's (1995) system, which makes use of 66 features. Unfortunately, Aone and Bennett do not report the features employed by their system.

is whether a small feature set encodes sufficient knowledge for training a high-performance coreference classifier.

It should be noted, though, that much work has been done on investigating the knowledge sources that are important for both pronoun resolution (e.g., Lappin and Leass (1994)) and coreference resolution (e.g., Grishman (1995), Lin (1995)). However, the investigation has proceeded to a large extent in the context of knowledge-based approaches to coreference resolution, which implies that linguistic constraints are generally treated as broadly and unconditionally applicable hard constraints. Furthermore, it is difficult to evaluate the relative contribution of the available knowledge sources in knowledge-based systems.

As a result, we investigate the question of whether increasing the *number* and *sophistication* of the features can improve the performance of a learning-based coreference system. In particular, we employ an arguably deep set of 57 features to represent a coreference instance $inst_{(NP_i,NP_j)}$ created from NP_j and one of its preceding noun phrases, NP_i. The features are partly derived from the ones used in existing coreference systems and partly chosen based on our linguistic intuition regarding coreference.

In the rest of this section, we will examine our 57 features in detail. Linguistically, the features can be divided into five groups: lexical, grammatical, semantic, positional, and knowledge-based. As we will see, each feature is either relational or non-relational. Non-relational features test some property P of one of NP_i and NP_j and take on a value of **Yes** or **No**, depending on whether P holds. Relational features test whether some property P holds for the NP pair under consideration and indicate whether the NPs are **Compatible** or **Incompatible** with respect to P; a value of **Not Applicable** is used when property P does not apply.

Lexical features. The lexical features are computed by performing string matching operations on NP_i and NP_j. Below we describe the 11 lexical features employed by our system.

L1: SOON_STR

The feature value is:

C if, after discarding determiners, the string denoting NP_i matches that of NP_j

I otherwise

Features L2-L5 restrict string matching to specific types of NPs. These features might give the learning algorithm additional flexibility in creating coreference rules on top of SOON_STR, since exact string match is likely to be a better coreference predictor for proper names than it is for pronouns, for instance.

L2: PRO_STR

This feature restricts exact string matching to pronouns. The feature value is:

C if both NPs are pronominal and are the same string

I otherwise

L3: PN_STR

This feature restricts exact string matching to proper names. The feature value is:

C if both NPs are proper names and are the same string

I otherwise

L4: WORDS_STR

This feature restricts exact string matching to non-pronominal noun phrases. The feature value is:

C if both NPs are non-pronominal and are the same string

I otherwise

L5: SOON_STR_NONPRO

This feature is essentially the same as SOON_STR, but restricts string matching to non-pronominal NPs. The feature value is:

C if both NPs are non-pronominal and after discarding determiners, the string of NP_i matches that of NP_j

I otherwise

In features L6 and L7, we perform string matching between word subsets of the two NPs.

L6: WORD_OVERLAP

This feature tests whether there is any overlap between the content words of the two NPs. The feature value is:

C if the intersection between the content words in NP_i and NP_j is not empty

I otherwise

L7: MODIFIER

This feature tests whether the modifiers of one NP are a subset of those of the other NP. The feature value is:

C if the prenominal modifiers of one NP are a subset of the prenominal modifiers of the other

I otherwise

In features L8 and L9, we check whether one NP is a substring of the other.

L8: PN_SUBSTR

This feature restricts substring matching to proper names. The feature value is:

C if both NPs are proper names and one NP is a proper substring (with respect to content words only) of the other

I otherwise

L9: WORDS_SUBSTR

This feature restricts substring matching to non-pronominal NPs. The feature value is:

C if both NPs are non-pronominal and one NP is a proper substring (with respect to content words only) of the other

I otherwise

Features L10 and L11 were originally proposed by Strube et al. (2002). The motivation is that most anaphors do not share the same (sub)string with their antecedents, and hence it is desirable to have a weakened version of the features for exact string match or substring match. As a result, Strube et al. compute the minimum edit distance (MED) between two NPs, NP_i and NP_j. Specifically, let m be the length of NP_i. The MED from NP_i to NP_j is equal to $\frac{m-(s+i+d)}{m}$, where s, i, and d are the minimum number of substitutions, insertions, and deletions needed to transform NP_i into NP_j (see Kruskal (1999) for details). Note that MED is an assymetric measure between the two NPs involved. As a result, we can compute MED for an NP pair in both directions, yielding the two features below.

L10: ANTE_MED

This feature measures the MED from NP_i to NP_j. The feature value is:

The minimum edit distance from NP_i to NP_j

L11: ANA_MED

This feature measures the MED from NP_j to NP_i. The feature value is:

The minimum edit distance from NP_j to NP_i

Grammatical features. Grammatical features test the grammatical properties of one or both NPs in the pair, and can be divided into five categories.

The first category of features determines the NP type of one or both of the noun phrases involved. The incorporation of these features is motivated by the fact that coreference strategies usually differ depending on the type of NPs.

G1: PRONOUN_1

This feature tests whether NP_i is a pronoun. The feature value is:

Y if NP_i is a pronoun

N otherwise

G2: PRONOUN_2

This feature tests whether NP_j is a pronoun. The feature value is:

Y if NP_j is a pronoun

N otherwise

G3: DEFINITE_1

This feature tests whether NP_i is a definite NP. The feature value is:

Y if NP_i starts with *the*

N otherwise

G4: DEFINITE_2

This feature tests whether NP_j is a definite NP. The feature value is:

Y if NP$_j$ starts with *the*

N otherwise

G5: DEMONSTRATIVE_2

This feature tests whether NP$_j$ starts with a demonstrative. The feature value is:

Y NP$_j$ starts with a demonstrative such as *this*, *that*, *these*, or *those*

N otherwise

G6: EMBEDDED_1

This feature tests whether NP$_i$ is a nested/embedded noun or NP. The feature value is:

Y if NP$_i$ is an nested/embedded noun or NP

N otherwise

G7: EMBEDDED_2

This feature tests whether NP$_j$ is a nested/embedded noun or NP. The feature value is:

Y if NP$_j$ is an nested/embedded noun

N otherwise

G8: IN_QUOTE_1

This feature tests whether NP$_i$ is part of a quoted string. The feature value is:

Y if NP$_i$ is part of a quoted string

N otherwise

G9: IN_QUOTE_2

This feature tests whether NP$_j$ is part of a quoted string. The feature value is:

Y if NP$_j$ is part of a quoted string

N otherwise

G10: BOTH_PROPER_NOUNS

This feature tests whether both NPs are proper names. The feature value is:

C if both NPs are proper names

NA if exactly one NP is a proper name

I otherwise

G11: BOTH_DEFINITES

This feature tests whether both NPs are definites. The feature value is:

C if both NPs start with *the*

I if neither start with *the*

NA otherwise

G12: BOTH_EMBEDDED

This feature tests whether both NPs are prenominal modifiers. The feature value is:

C if both NPs are prenominal modifiers

I if neither are prenominal modifiers

NA otherwise

G13: BOTH_IN_QUOTES

This feature tests whether both NPs are in some quoted strings in the associated text. The feature value is:

C if both NPs are part of a quoted string

I if neither are part of a quoted string

NA otherwise

G14: BOTH_PRONOUNS

This feature tests whether both NPs are pronouns. The feature value is:

C if both NPs are pronouns

I if neither are pronouns

NA otherwise

The second category of features tests the grammatical role of one or both of the noun phrases involved. These features are motivated by the observation that NPs in salient grammatical positions (such as subjects) are more likely to be antecedents than those that are not (e.g., Lappin and Leass (1994), Mitkov (1998)). In our implementation, information on the grammatical role of an NP is provided by Lin's (1998) MINIPAR dependency parser.

G15: BOTH_SUBJECTS

This feature tests whether both NPs are grammatical subjects. The feature value is:

C if both NPs are grammatical subjects

I if neither are subjects

NA otherwise

G16: SUBJECT_1

This feature tests whether NP_i is a grammatical subject. The feature value is:

Y if NP_i is a subject

N otherwise

G17: SUBJECT_2

This feature tests whether NP_j is a grammatical subject. The feature value is:

Y if NP_j is a subject

N otherwise

G18: GRAMROLE_1

This feature encodes the grammatical role of NP_i. The feature value is:

The grammatical role of NP_i

G19: GRAMROLE_2

This feature encodes the grammatical role of NP_j. The feature value is:

The grammatical role of NP_j

The third category of features encodes traditional linguistic constraints on when two NPs can or cannot be coreferent. As mentioned above, previous work (e.g., Grishman (1995), Lin (1995)) generally treats linguistic constraints as broadly and unconditionally applicable hard constraints. Because sources of linguistic information are represented as features in our system, we can incorporate them selectively rather than as universal hard constraints.

G20: NUMBER

This feature tests for number agreement between the two NPs. The feature value is:

C	if NP_i and NP_j agree in number
I	if they disagree in number
NA	if number information for one or both NPs cannot be determined

G21: GENDER

This feature tests for gender agreement between the two NPs. The feature value is:

C if NP_i and NP_j agree in gender

I if they disagree in gender

NA if gender information for one or both NPs cannot be determined

G22: APPOSITIVE

This feature tests whether the two NPs form an **appositive** construction. An appositive is an NP that is set beside another NP to explain or identify it. In our *Queen Elizabeth* example in Figure 1, *husband* and *King George VI* form an appositive construction, with *King George VI* being the appositive of *husband*. The feature value is:

C if the NPs are in an appositive relationship

I otherwise

G23: AGREEMENT

This feature tests for number and gender agreement between the two NPs. The feature value is:

C if the NPs agree in both gender and number

I if they disagree in either gender or number

NA if number or gender information for one or both NPs cannot be determined

G24: ANIMACY

This feature tests whether the two NPs match in animacy. The feature value is:

C if the NPs match in animacy (i.e., both are animate or both are not)

I if the NPs do not match in animacy

NA if animacy information for one or both NPs cannot be determined

G25: MAXIMALNP

This feature tests whether two NPs have the same maximal NP projection, im-

plicitly encoding the constraint that the two NPs cannot be coreferent if they do. The feature value is:

I if both NPs have the same maximal NP projection

C otherwise

G26: PREDNOM

This feature tests whether the two NPs are in a **predicate nominal** construction. Two NPs form a predicate nominal construction if one NP is predicative of the other NP. So, for instance, in the sentence *King George VI is Queen Elizabeth's husband*, the two NPs *King George VI* and *husband* form a predicate nominal construction. The feature value is:

C if the NPs form a predicate nominal construction

I otherwise

G27: SPAN

This feature encodes the constraint that two NPs cannot be coreferent if one spans the other. Specifically, NP_i spans NP_j if the text span of NP_j covers (and is not identical) that of NP_i. For instance, *Microsoft executives* spans *Microsoft* if their text spans overlap.[5] The feature value is:

I if one NP spans the other

C otherwise

G28: BINDING

This feature approximates the output specified by conditions B and C of the **binding theory** (see Chomsky (1988) for an introduction). In essence, the theory is

[5]Note that *Queen Elizabeth's husband* spans *husband* if their text spans overlap; if they do, they are essentially the *same* NP and only *Queen Elizabeth's husband* will be extracted from the text by our NP extraction algorithm.

comprised of three conditions (A, B, and C) that specify when two NPs can or cannot co-refer based purely on syntactic configurations. (In our implementation, syntactic information is automatically derived from parse trees constructed by Charniak's (2000) statistical parser.) In particular, conditions B and C impose constraints on the interpretation of pronouns and referential expressions, respectively.

I if the NPs violate conditions B or C of the binding theory

C otherwise

G29: CONTRAINDICES

This feature encodes the intra-sentential contraindexing constraints that (1) two NPs separated by a preposition cannot be co-indexed and (2) two non-pronominal NPs separated by a non-copular verb cannot be co-indexed. The feature value is:

I if the NPs cannot be co-indexed based on the above heuristics

C otherwise

G30: SYNTAX

This feature is basically a conjunction of several primitive grammatical features described above. The feature value is:

I if the NPs have incompatible values for the BINDING, CONTRAINDICES, SPAN or MAXIMALNP constraints

C otherwise

G31: PRO_EQUIV

This feature encodes a very simple heuristic for performing pronoun resolution. The feature value is:

C if both NPs are pronouns, agree in both gender and number, and appear in consecutive sentences

I otherwise

The fourth category encodes linguistic preferences either for or against coreference.

G32: INDEFINITE

This feature encodes the observation that an indefinite NP that is not an appositive is not likely to be anaphoric. The feature value is:

Y if NP_j is an indefinite and is not an appositive

N otherwise

G33: PRONOUN

This feature encodes the fact that pronominal antecedents for non-pronominal NPs are discouraged. The feature value is:

I if NP_i is a pronoun and NP_j is not

C otherwise

The last category of features essentially encode slightly more complex, but generally non-linguistic heuristics.

G34: CONSTRAINTS

This feature is a conjunction of some of the features described above. The feature value is:

C if the NPs agree in GENDER and NUMBER and do not have incompatible values for CONTRAINDICES, SPAN, ANIMACY, PRONOUN, and CONTAINS_PN

I if the NPs have incompatible values for any of the above features

NA otherwise

G35: CONTAINS_PN

This feature effectively disallows coreference between NPs that contain distinct proper names but are not themselves proper names (e.g., "IBM executives" and "Microsoft executives"). The feature value is:

I if both NPs are not proper names but contain proper names that mismatch on every word

C otherwise

G36: PROPER_NOUN

This feature effectively disallows coreference between two distinct proper names. The feature value is:

I if both NPs are proper names, but mismatch on every word

C otherwise

G37: TITLE

This feature disallows coreference between two titles. Here, we only consider titles that are descriptors immediately preceding the name of a person (e.g., the descriptor *Princess* in *Princess Diana*). The feature attempts to prevent coreference between the occurrences of *Princess* in the two NPs *Princess Diana* and *Princess Margaret*, for instance. The feature value is:

I if one or both of the NPs is a title of a person

C otherwise

Semantic features. Our system has six features for performing various semantic compatibility tests between two NPs, as described below.

S1: WNCLASS

This feature tests the whether the two NPs have the same semantic class according to the WordNet semantic knowledge base (Fellbaum, 1998) as well as a named entity identifier (Bikel et al., 1999) for identifying pre-defined semantic classes such as person, organization, location, date, time, money, and percent.[6] The feature value is:

C if the NPs have the same semantic class

I if they do not have the same semantic class

NA if the semantic class information for one or both NPs cannot be determined

S2: ALIAS

This feature tests whether one NP is a name alias of the other. The implementation if this feature follows the description of the ALIAS feature in the Soon et al. (2001) system. Specifically, different mechanisms are used to test for name aliases for NPs of different semantic classes. If both NPs are dates, we first normalize them by a date tagger (Mani and Wilson, 2000) and then check whether the two refer to the same date. If both are persons (e.g., *John Smith* and *Smith*), we pick the last word of each NP and check whether the two words are the same. If both are organizations (e.g., *General Electric* and *GE*), we check whether one is a potential acronym of the other. The feature value is:

[6] A precise definition of these named entity classes can be found in the documentation of the MUC-7 named entity task. See `http://www.itl.nist.gov/iad/894.02/related_projects/muc/proceedings/ne_task.html`.

C	if one NP is an alias of the other
I	otherwise

S3: CLOSEST_COMP

This feature attempts to identify for NP_j the closest preceding NP that is semantically compatible with it. The feature value is:

C	if NP_i is the closest NP preceding NP_j that has the same semantic class as NP_j and the two NPs do not violate any of the linguistic constraints
I	otherwise

S4: SUBCLASS

This feature tests for an ancestor-descendent relationship in WordNet. The feature value is:

C	if (1) the NPs have different head nouns, (2) the two head nouns exist in WordNet and (3) the head nouns have an ancestor-descendent relationship in WordNet using any of the available word senses
I	otherwise

S5: WNDIST

This feature measures the WordNet graph-traversal distance between the two NPs. The feature value is:

Distance between NP_i and NP_j in WordNet (using the first word sense only)	if NP_i and NP_j have an ancestor-descendent relationship but have different heads
Infinity	otherwise

S6: WNSENSE

With this feature, we can check whether the WordNet sense number is useful for

predicting coreference relationships. The feature value is:

The smallest sense number in WordNet for which there exists an ancestor-descendent relationship between the two NPs	if NP_i and NP_j have different heads
Infinity	otherwise

Positional features. Our two positional features measure the textual distance between the two NPs in different ways, as shown below.

P1: SENTNUM

This feature measures the distance between the NPs in terms of the number of sentences. The feature value is:

The distance between the NPs in terms of the number of sentences.

P2: PARANUM

This feature measures the distance between the two NPs in terms of the number of paragraphs. The feature value is:

The distance between the NPs in terms of the number of paragraphs.

Knowledge-based features. Our system has one knowledge-based feature, as shown below.

K1: PRO_RESOLVE

This feature encodes the output of an in-house naive pronoun resolution algorithm that closely follows the design of Baldwin's (1997) CogNIAC system. The feature value is:

C if NP_j is a pronoun and NP_i is its antecedent according to the naive pronoun resolution algorithm

I otherwise

4.6 Anaphoricity Determination

Our second linguistic extension is concerned with anaphoricity determination. As seen in Section 2.1, the goal of anaphoricity determination is to decide whether a discourse entity is anaphoric or not. Non-anaphoric entities, by definition, do not possess an antecedent and hence the clustering algorithm will not need to search for an antecedent for these entities if anaphoricity information is available. This observation naturally motivates a new system architecture for coreference resolution that employs an explicit anaphoricity determination component as a pre-processing filter for the coreference system.

Given the potential usefulness of knowledge of (non-)anaphoricity for coreference resolution, anaphoricity determination has been studied fairly extensively (e.g., Paice and Husk (1987), Lappin and Leass (1994), Kennedy and Boguraev (1996), Denber (1998), Bean and Riloff (1999), Vieira and Poesio (2000a), Evans (2001)). Interestingly, existing machine learning approaches to coreference resolution have performed reasonably well without anaphoricity determination (e.g., Aone and Bennett (1995), McCarthy and Lehnert (1995), Soon et al. (2001), Yang et al. (2003)). Nevertheless, there is empirical evidence that resolution systems can further be improved with anaphoricity information. For instance, Strube and Müller (2003) point out that the accuracy of their pronoun resolution system, which does not employ anaphoricity determination, is severely limited by the fact that an antecedent is mistakenly found for many non-anaphoric pronouns. This

motivates our work of improving learning-based coreference systems using automatically computed anaphoricity information.

In this section, we propose a new supervised learning approach to anaphoricity determination. Our approach combines the following advantages that together distinguish it from existing approaches to the problem.

- Since the approach is supervised, no hand-coded heuristics are required.
- All of the features accessible to the learning algorithm (and hence the resulting classifier) are domain-independent.
- Examining the resulting classifier allows additional insight into the factors important to anaphoricity determination from an information-theoretic perspective.
- The approach performs anaphoricity determination on *all* types of NPs, including pronouns, proper NPs, and common NPs.
- The approach performs *global optimization*, in which the anaphoricity determination procedure is explicitly maximized for coreference performance.

Comparison with related work. Among the five features of our approach described above, global optimization is the characteristic that uniquely distinguishes it from existing approaches. To our knowledge, all of the existing approaches to anaphoricity determination perform *local optimization*, in which the anaphoricity determination procedure is developed and optimized independently of the coreference system that uses the computed anaphoricity determination. Though such modular design is appealing from a software engineering perspective, local opti-

Table 4.1: Comparison of related work on anaphoricity determination

Systems	Targeted NP type	Knowledge-based or learning-based?
Paice and Husk (1987)	pleonastic pronouns	knowledge-based
Lappin and Leass (1994)	pleonastic pronouns	knowledge-based
Kennedy and Boguraev (1996)	pleonastic pronouns	knowledge-based
Denber (1998)	pleonastic pronouns	knowledge-based
Bean and Riloff (1999)	definite NPs	unsupervised learning
Vieira and Poesio (2000a)	definite NPs	knowledge-based
Evans (2001)	all uses of *it*	supervised learning

mization may yield a set of anaphoricity decisions that are sub-optimal with respect to improving coreference systems.

As mentioned above, the application of machine learning techniques to the problem as well as the coverage of all types of noun phrases are also major features of our approach. Table 4.1 compares related work along these two dimensions: what types of NPs the approach can handle, and whether the approach is knowledge-based or learning-based.

As we can see, none of the existing approaches attempt to perform anaphoricity determination on all types of NPs. Some address only pleonastic pronouns (e.g., Paice and Husk (1987), Lappin and Leass (1994), Kennedy and Boguraev (1996), Denber (1998)), while others focus on identifying anaphoric and non-anaphoric uses of *it* (e.g., Evans (2001)) and definite descriptions (e.g., Bean and Riloff (1999), Vieira and Poesio (2000a)). For coreference resolution, however, it is crucial that

all types of NPs be handled by an anaphoricity determination procedure.

In addition, unlike our approach, most existing approaches are knowledge-based, relying on a set of hand-crafted heuristics for distinguishing anaphoric and non-anaphoric NPs. Perhaps the only exceptions are the work of Bean and Riloff (1999) and Evans (2001). Bean and Riloff (1999), however, focus on an *unsupervised* approach for constructing a list of non-anaphoric entities from an unannotated corpus. Although their approach is domain-independent, the strength of the approach lies in the use of lexical information for acquiring a list of non-anaphoric entities consisting mainly of domain-specific terms, which is useful for classifying documents from that domain. Evans' (2001) work is similar to ours in that both adopt a *supervised* approach to the problem. However, as shown in Table 4.1, Evans' work handles only anaphoric and non-anaphoric uses of the pronoun *it*.

In the rest of this section, we will describe our globally-optimized, supervised approach to anaphoricity determination in detail. Our discussion will proceed in two steps. We will first examine how to construct a locally-optimized anaphoricity classifier in Section 4.6.1. Then, in Section 4.6.2, we will show how to generalize this local approach to a global approach.

4.6.1 The Locally-Optimized Approach

As mentioned above, we will describe how to construct a locally-optimized anaphoricity classifier in this subsection.

Building a classifier for anaphoricity determination. A machine learning algorithm is used to train a classifier that, given a description of an NP in a document, NP_j, decides whether or not the NP is anaphoric. Each training instance

represents a single NP and consists of 37 features that are potentially useful for distinguishing anaphoric and non-anaphoric NPs. Linguistically, the features can be divided into four groups: lexical, grammatical, semantic, and positional.

Lexical features. The lexical features test whether some property holds for an NP based on its corresponding surface string. We employ four lexical features in training an anaphoricity classifier, as described below. (Note: although some of the anaphoricity features share the same name as the coreference features described in Section 4.4, their definitions may not be identical.)

AL1: STR_MATCH

This feature encodes the preference that NP_j is likely to be anaphoric if some preceding NP shares the same string with it after determiners are discarded. The feature value is:

Y	if there exists an NP NP_i preceding NP_j such that, after discarding determiners, NP_i and NP_j are the same string
N	otherwise

AL2: HEAD_MATCH

This feature encodes the preference that NP_j is likely to be anaphoric if some preceding NP shares the same head with it after determiners are discarded. The feature value is:

Y	if there exists an NP NP_i preceding NP_j such that NP_i and NP_j have the same head
N	otherwise

AL3: UPPERCASE

This feature tests whether the NP is entirely in uppercase. The feature value is:

Y if NP$_j$ is entirely in uppercase

N otherwise

AL4: CONJ

This feature tests whether the NP is a conjunction. The feature value is:

Y if NP$_j$ is a conjunction

N otherwise

Grammatical features. The grammatical features can be subcategorized into three groups.

The first group simply determines the NP type, as shown below. Intuitively, indefinite NPs, possessives, and bare plural NPs are less likely to be anaphoric than definite NPs, for instance.

AG1: DEFINITE

This feature tests whether the NP is definite. The feature value is:

Y if NP$_j$ starts with *the*

N otherwise

AG2: DEMONSTRATIVE

This feature tests whether the NP starts with a demonstrative. The feature value is:

Y if NP$_j$ starts with a demonstrative such as *this*, *that*, *these*, or *those*

N otherwise

AG3: INDEFINITE

This feature tests whether the NP is an indefinite. The feature value is:

Y if NP$_j$ starts with *a* or *an*

N otherwise

AG4: QUANTIFIED

This feature tests whether the NP is a quantified NP. The feature value is:

Y if NP$_j$ starts with quantifiers such as *every*, *some*, *all*, *most*, *many*, *much*, *few*, and *none*

N otherwise

AG5: ARTICLE

This feature combines the DEFINITE, INDEFINITE, and QUANTIFIED features described above. The feature value is:

DEFINITE if NP$_j$ is a definite NP

QUANTIFIED if NP$_j$ is a quantified NP

INDEFINITE if NP$_j$ is an indefinite NP

UNKNOWN otherwise

AG6: PRONOUN

This feature determines the case of a pronominal NP. The feature value is:

NOMINATIVE if NP$_j$ is pronominal and has nominative case

ACCUSATIVE if NP$_j$ is pronominal and has accusative case

POSSESSIVE if NP$_j$ is a possessive pronoun

PLEONASTIC if NP$_j$ is a pleonastic pronoun

NONE if NP$_j$ is not a pronoun

UNKNOWN otherwise

AG7: PROPER_NOUN

This feature tests whether the NP is a proper noun. The feature value is:

Y if NP_j is a proper noun

N otherwise

AG8: POSSESSIVE

This feature tests whether the NP is preceded by a possessive NP. The feature value is:

Y if NP_j starts with or is immediately preceded by a possessive pronoun or NP

N otherwise

AG9: BARE_SINGULAR

This feature tests whether the NP is a bare singular NP. The feature value is:

Y if NP_j is singular and does not start with an article

N otherwise

AG10: BARE_PLURAL

This feature tests whether the NP is a bare plural NP. The feature value is:

Y if NP_j is plural and does not start with an article

N otherwise

AG11: EMBEDDED

This feature tests whether the NP is embedded/nested. The feature value is:

Y if NP_j is a prenominal modifier

N otherwise

The second group of grammatical features tests some syntactic property of an NP. Noun phrases in certain syntactic constructions, including predicated NPs, NPs that have an appositive, and NPs that are postmodified by a relative clause, indicate *unfamiliar* uses and are not likely to be anaphoric. Typically, unfamiliar uses arise in situations where the interpretation of an NP depends on additional

information attached to the NP. This set of features aims to identify (un)familiar uses of an NP.

AG12: APPOSITIVE

This feature tests whether the NP has an appositive. The feature value is:

Y	if NP_j is the first of the two NPs in an appositive construction
N	otherwise

AG13: PREDNOM

This feature tests whether the NP is predicated by another NP. The feature value is:

Y	if NP_j is the first of the two NPs in a predicate nominal construction
N	otherwise

AG14: NUMBER

This feature encodes the number of the NP. The feature value is:

SINGULAR	if NP_j is singular in number
PLURAL	if NP_j is plural in number
UNKNOWN	if the number information cannot be determined

AG15: MODIFIER

This feature tests whether the NP is premodified. The feature value is:

Y	if NP_j is premodified
N	otherwise

AG16: POSTMODIFIED

This feature tests whether the NP is postmodified. The feature value is:

Y if NP$_j$ is postmodified by a relative clause

N otherwise

AG17: CONTAINS_PN

This feature tests whether the NP contains a proper noun but is not itself a proper noun (e.g., *IBM executive*). The feature value is:

Y if NP$_j$ is not a proper noun but contains a proper noun

N otherwise

AG18: SPECIAL_NOUNS

Certain nouns, referred to as **special nouns** by Vieira and Poesio (2000b), are not likely to be anaphoric because they refer to unfamiliar uses (e.g., *fact*, *result*, *conclusion*) or to entities whose existence is of commonsense knowledge (e.g., time references such as *year*, *day*, *week*, *month*, *hour*, *time*, *morning*, *afternoon*, *night*, *period*, *quarter* and their plurals). At the same time, certain modifiers can make an NP not likely to be anaphoric. Examples include Hawkins' (1978) **unexplanatory modifiers** (e.g., *first*, *last*, *best*, *most*, *maximum*, *minimum*, *only*, *closest*, *greatest*, *biggest*), superlatives, and certain relatives (e.g., *more*, *closer*, *bigger*, *greater*). We create a feature using this set of words. The feature value is:

Y if NP$_j$'s head noun is a special noun or its modifier is an unexplanatory modifier

N otherwise

Finally, the "syntactic pattern" features, largely created based on a combination of words and part-of-speech tags, generally identify non-anaphoric definite NPs.

AG19: THE_N

The feature value is:

Y if NP_j consists of *the <common noun>*

N otherwise

AG20: THE_2N

The feature value is:

Y if NP_j consists of *the <common noun><common noun>*

N otherwise

AG21: THE_PN

The feature value is:

Y if NP_j consists of *the <proper noun>*

N otherwise

AG22: THE_PN_N

The feature value is:

Y if NP_j consists of *the <proper noun><common noun>*

N otherwise

AG23: THE_ADJ_N

The feature value is:

Y if NP_j consists of *the <adjective><common noun>*

N otherwise

AG24: THE_NUM_N

The feature value is:

Y if NP_j consists of *the <cardinal><common noun>*

N otherwise

AG25: THE_NE

The feature value is:

Y if NP_j consists of *the* immediately followed by a named entity

N otherwise

AG26: THE_SING_N

The feature value is:

Y if NP_j consists of *the* immediately followed by a singular NP not containing any proper noun

N otherwise

Semantic features. Instances also encode four semantic features, as described below.

AS1: ALIAS

This feature encodes the preference that NP_j is likely to be anaphoric if some preceding NP is its name alias. Name aliases are identified in the same way as in feature S2. The feature value is:

Y if there exists an NP NP_i preceding NP_j such that NP_i and NP_j are aliases

N otherwise

AS2: POST

This feature encodes the preference that NPs that are titles are not likely to be anaphoric. By post we mean a job title that is realized as a descriptor immediately preceding the name of a person (e.g., *Chief Executive Officer* in *Chief Executive Officer Larry Ellison*). The feature value is:

Y if NP_j is a post

N otherwise

AS3: SUBCLASS

This feature encodes the constraint that an NP cannot be anaphoric if there does not exist a preceding NP that is semantically compatible with it. The feature value is:

Y	if there exists an NP NP_i preceding NP_j such that (1) the two NPs have different head nouns, (2) the two head nouns exist in WordNet, and (3) the head nouns have an ancestor-descendent relationship in WordNet using any of the available word senses
N	otherwise

AS4: TITLE

This feature encodes the preference that NPs that are titles are not likely to be anaphoric. Here, we only consider titles that are descriptors immediately preceding the name of a person (e.g., the descriptor *Princess* in *Princess Diana*). The feature value is:

Y	if NP_j is the title of a person
N	otherwise

Positional features. The final set of features makes anaphoricity decisions based on the position of the NP in the text. This set of features is motivated by the observation that NPs that appear towards the beginning of a text are less likely to be anaphoric than those that appear later in the text.

AP1: FIRST_SENT

This feature tests whether the NP appears in the first sentence of a text. The feature value is:

Y if NP_j is in the first sentence of the body of the text

N otherwise

AP2: FIRST_PARA

This NP tests whether the NP appears in the first paragraph of a text. The feature value is:

Y if NP_j is in the first paragraph of the body of the text

N otherwise

AP3: HEADER

This feature tests whether the NP appears in the header portion of a text. The feature value is:

Y if NP_j is in the header of the text

N otherwise

As mentioned above, each instance is represented by these 37 features. Now, the classification associated with a training instance — one of ANAPHORIC or NOT ANAPHORIC — is obtained from coreference chains in the training texts. Specifically, a *positive instance* is created for each NP that is involved in a coreference chain but is not the head of the chain. A *negative instance* is created for each of the remaining NPs.

Applying the classifier. To determine the anaphoricity of an NP in a test document, we create an instance for it as during training and present it to the decision tree anaphoricity classifier, which returns a value of ANAPHORIC or NOT ANAPHORIC.

4.6.2 The Globally-Optimized Approach

As mentioned above, locally-optimized anaphoricity determination procedures suffer from the potential problem that the resulting set of anaphoricity decisions may be sub-optimal with respect to improving coreference systems. To address this problem, we propose a new approach to anaphoricity determination that explicitly maximizes for coreference performance. Our globally-optimized approach, as we will see, is a simple generalization of the locally-optimized approach described in the previous subsection.

The central idea behind the globally-optimized approach is to view anaphoricity determination as a problem of determining how *conservative* an anaphoricity classifier should be in classifying an NP as (non-)anaphoric. Recall that the coreference system will only search for an antecedent for NPs that are determined to be anaphoric by the anaphoricity determination component. Hence, if the classifier is too liberal in classifying an NP as non-anaphoric, then many anaphoric NPs will be misclassified, ultimately leading to a deterioration of recall and of the overall performance of the coreference system. On the other hand, if the classifier is too conservative, then only a small fraction of the truly non-anaphoric NPs will be identified, and so the resulting anaphoricity information may not be effective in improving the coreference system. The challenge then is to determine a "good" degree of conservativeness.

Before we show how to adjust the conservativeness of a classifier, let us examine a parameter present in many off-the-shelf machine learning algorithms for training a classifier — the cost ratio (cr), which is defined as follows.

$$cr := \frac{\text{cost of misclassifying a positive instance}}{\text{cost of misclassifying a negative instance}}$$

The *cr* parameter provides a means of adjusting the relative misclassification penalties placed on training instances of different classes. In particular, the larger *cr* is, the more conservative the resulting classifier is in classifying an instance as negative. Given this observation, we can naturally define the conservativeness of an anaphoricity classifier as follows. We say that classifier A is more conservative than classifier B in determining an NP as non-anaphoric if A is trained with a higher cost ratio than B.

Now, to achieve global optimization, we can simply tune *cr* to optimize for coreference performance on held-out data.

It should be easy to see that the locally-optimized approach to anaphoricity determination is a special case of the global one. Unlike the global approach in which the conservativeness parameter is tuned based on labeled data, the local approach uses "default" parameter values. For most learning algorithms that provide the option of adjusting the cost ratio, the default value of *cr* is one.

Finally, note that any learning algorithm that provides *cr* as a training parameter can be used to train our globally-optimized anaphoricity classifier. We will use RIPPER for this purpose.

4.7 Putting Everything Together

Given the extensions we have proposed thus far, a number of design decisions have to be made in order to integrate them into our coreference systems. Specifically:

- In Section 4.1, we saw three clustering algorithms, namely closest-first clustering, best-first clustering, and aggressive-merge clustering. Which clustering algorithm should we use?

- In Section 4.2, we saw three training instance creation methods by McCarthy and Lehnert, Aone and Bennett, and Soon et al. In addition, we proposed a parametric family of instance creation methods based on a generalization of Soon et al.'s method. Which of these methods should we use?

- In Section 4.3, we proposed an algorithm for pruning the learned coreference ruleset so that clustering performance is explicitly maximized. Although we hypothesized that rule pruning would boost coreference performance, this may not be the case in practice. Should we incorporate rule pruning or not?

- In Section 4.4, we mentioned two learning algorithms, the RIPPER rule-learning algorithm and the C4.5 decision tree induction system. Which learner should we use?

- In Section 4.5, we suggested a large-scale expansion in the number and sophistication of the features available to the learning algorithm for training a coreference classifier. The expanded feature set can potentially increase performance because of the availability of additional knowledge. However, performance may deteriorate if irrelevant features were present (see Blum and Langley (1997)). The question then is: should we learn from our expanded feature set (possibly with feature selection to discard irrelevant features), or should we resort to one of the previously proposed feature sets?

- In Section 4.6, we proposed a supervised approach to anaphoricity determination that explicitly optimizes for coreference performance. Anaphoricity determination is expected to help improve system performance. But like rule pruning, this may not be the case in practice. Should we incorporate anaphoricity determination or not?

TuneCorefParams(T, S, Λ)

Input: A training corpus T, a scoring function S, and a parametric coreference system Λ

Algorithm:
Randomly divide T into two disjoint sets A and B such that A is twice as large as B in terms of the number of documents.

foreach possible coreference parameter setting $\lambda \in \Lambda$ **do**
  Train the underlying coreference and anaphoricity classifiers on A given λ.
  $Score(\lambda)$:= score of the resulting coreference system on B according to S.

$\lambda^* := \arg\max_{\lambda \in \Lambda} Score(\lambda)$

Output: λ^*

Figure 4.4: The **TuneCorefParams** Algorithm

We can approach all of these questions simultaneously by viewing each of them as a distinct parameter to our coreference system. Our goal would then be to find a parameter setting that can yield good coreference system performance.

Now, recall that one of our criticisms of existing learning-based coreference systems is their reliance on ad-hoc design decisions. To avoid ad-hoc decision making as much as possible, we desire an *automatic* parameter computation process. This suggests a *corpus-based* approach, in which the parameters are tuned so that coreference system performance is maximized on held-out development data.

Our parameter tuning algorithm, **TuneCorefParams**, is shown in Figure 4.4. **TuneCorefParams** takes as input a training corpus T, a coreference scoring function S, and a parametric coreference system Λ (with one parameter corresponding to each of the above questions). The algorithm first randomly divides the docu-

ments in T into two sets A and B, using $\frac{2}{3}$ of the documents for training purposes (A) and reserving the rest for development (B). Then it exhaustively enumerates all possible parameter settings. Specifically, for each parameter setting λ, **TuneCorefParams** trains the underlying coreference and anaphoricity classifiers on A given λ and evaluates the resulting system on B. Finally, the parameter setting λ^* that yields the best performance on B according to S is returned (ties are broken by giving preference to systems with higher precision), and the coreference system defined by λ^* is selected for evaluation on unseen data.

TuneCorefParams exhibits a number of potential advantages of our approach to building a learning-based coreference system over existing approaches:

- The model parameters are computed automatically, thus avoiding potentially ad-hoc design decisions.
- The model parameters can be re-computed for each new coreference data set as long as labeled training data is available. This allows us to easily configure and optimize the coreference system for a particular corpus.
- The algorithm ensures that the parameters of the coreference system are optimized with respect to the given coreference scoring function. Furthermore, parameterizing the algorithm with the scoring function allows us to optimize our coreference system with respect to *any* coreference scoring function.

Note, however, that since **TuneCorefParams** performs an exhaustive search for the optimal parameter setting, we have to ensure that the search space is finite. This implies that we need to constrain each dimension of the search space to be finite. Below we show how these constraints are imposed (if necessary) by considering each dimension in turn.

Clustering algorithm. Since we are considering only three clustering algorithms (closest-first, best-first, and aggressive-merge), this dimension is finite without imposing extra constraints.

Training instance creation. In this dimension, we are considering the training instance creation methods proposed by McCarthy and Lehnert, Aone and Bennett, and our parametric Soon-n method. To ensure that this dimension is finite, we only consider integral values of n within the range [1, 5].

Rule pruning. This dimension is finite without imposing extra constraints, since we are only considering whether rule pruning should be incorporated or not. Nevertheless, two points deserve mentioning. First, as mentioned before, we only consider applying rule pruning to RIPPER classifiers but not decision tree classifiers. Second, we reuse the training corpus for pruning purposes, due to the paucity of labeled training data for some of our coreference data sets, as we will see in the next chapter.

Learning algorithm. This dimension is finite without imposing extra constraints, since we are only deciding between RIPPER and C4.5.

Feature set. As mentioned above, the learning algorithm can be trained using either our expanded feature set (possibly with feature selection to discard irrelevant features) or one of the previously proposed feature sets. Preliminary experiments with forward selection and backward elimination (Kohavi and John, 1997) indicate that feature selection is computationally very expensive and yet is not effective in improving system performance. Ruling out the possibility of performing feature selection, we are basically left with two (and hence a finite number of) choices.

Table 4.2: Feature set for the Soon coreference system

Feature Type	Feature
Lexical	SOON_STR
Grammatical	PRONOUN_1, PRONOUN_2, DEFINITE_2, DEMONSTRATIVE_2, NUMBER, GENDER, BOTH_PROPER_NOUNS, APPOSITIVE
Semantic	WNCLASS, ALIAS
Positional	SENTNUM

Now, the question is which of the previously proposed feature sets should be considered in addition to our expanded feature set. Given that the Soon et al. system achieves reasonably good performance on the MUC data sets, we will use their feature set, which consists of 12 surface-level features. In fact, these 12 features (shown in Table 4.2) are a subset of our 57 features.

Anaphoricity determination. In this case, we are considering the *cr*-parameterized anaphoricity determination models. To ensure that this dimension is finite, we will only consider integral values of *cr* within the range [1, 10]. Preliminary experiments with all of our data sets indicate that this range is sufficiently large to cover the optimal value.

Computing the size of the search space. So far we have discussed how we constrain each dimension in the search space to be finite. These constraints are summarized in Table 4.3, where each parenthesized number in the first column indicates the size of the corresponding dimension. Given these constraints, we can easily count the number of distinct coreference system configurations in our (finite)

Table 4.3: Constraints on the allowable values of each parameter to the coreference system

Coreference System Parameter	Allowable Values
Clustering algorithm (3)	closest-first; best-first; aggressive-merge
Training instance creation method (7)	McCarthy and Lehnert; Aone and Bennett; Soon-n with $n = 1, \ldots, 5$
Rule pruning (2)	with or without rule pruning
Learning algorithm (2)	RIPPER; C4.5
Feature set (2)	the Soon feature set, our expanded feature set
Anaphoricity determination (11)	training anaphoricity classifiers with $cr = 1, \ldots, 10$; no anaphoricity determination

search space. Specifically, since we only apply rule pruning to the coreference classifiers trained with RIPPER, we have $3 \times 7 \times 2 \times 2 \times 11$ systems with RIPPER classifiers and $3 \times 7 \times 2 \times 11$ systems with decision tree classifiers. This yields a total of 1386 distinct coreference systems. The finite parameter search space guarantees that **TuneCorefParams** will terminate.

4.8 Chapter Summary

In this chapter, we proposed a set of extensions to the standard machine learning framework for coreference resolution, with the goal of improving existing machine learning approaches to the problem. We began our discussion with four extra-linguistic extensions involving the clustering algorithm, the training instance selection method, the learning algorithm, and the optimization of the automatically

acquired coreference ruleset. Then we examined two linguistic extensions, one involving anaphoricity determination and the other a large-scale expansion in the number and sophistication of features available to the learning algorithm. Finally, we presented an algorithm for integrating these extensions into our coreference system. The next chapter will be devoted to evaluating our coreference system.

CHAPTER 5

EVALUATION

The fact that one system was correct 50 percent of the time while another was correct only 40 percent of the time says nothing about the long-term viability of either approach. — Allen (1995)

In the previous chapter, we proposed a set of extensions to the standard machine learning framework for coreference resolution. When attempting to integrate all of these extensions into our coreference system, we noted that a design decision has to be made for each such extension. To automate the decision making process, and to ensure that design decisions are made collectively and not independently of each other, we proposed an algorithm in which coreference resolution is viewed as a parameter search problem. Specifically, each extension corresponds to a parameter of the coreference system, and the goal then is to search for a set of parameters that yields the best-performing coreference system on held-out data.

This chapter is devoted to evaluating the viability of the above approach to building a learning-based coreference resolution system. Sections 5.1 and 5.2 introduce the coreference corpora and the scoring program that we will use in our evaluation, respectively. Section 5.3 explains the steps we take to pre-process the documents in the coreference corpora. Finally, we present evaluation results in Section 5.4.

5.1 Coreference Corpora

In our evaluation, we employ two standard coreference corpora, MUC and ACE. Both corpora are named after the conferences for which they were created. MUC

(Message Understanding Conference) is a series of DARPA-sponsored conferences held to foster the development of information extraction technologies[1], whereas ACE (Automatic Content Extraction) is a NIST-sponsored program aiming to develop technologies to automatically extract and characterize meaning from natural language data.[2] In both MUC and ACE, coreference resolution is identified as a key information extraction task. To facilitate the evaluation of coreference systems, corpora made up of a set of articles manually annotated with coreference chains were created. The MUC corpus consists of the MUC-6 (MUC-6, 1995) and MUC-7 (MUC-7, 1998) data sets. The ACE corpus, on the other hand, is composed of three data sets made up of three different news sources: Broadcast News (BNEWS), Newspaper (NPAPER), and Newswire (NWIRE). Each of these five data sets is in turn comprised of a set of "dry run" texts primarily used for training and development as well as a set of "formal evaluation" texts for testing purposes.

Statistics collected from the five data sets are shown in Table 5.1. The first two concern the size of a data set as measured by the number of texts and the number of tokens, respectively. The third one is the number of NPs that are involved in a coreference relationship (i.e., the number of non-singletons) based on the annotated data. The last one is the *minimum* number of coreference links that need to be discovered by a coreference system in order to generate the correct NP partition for the text under consideration.[3] We also collect the same set of statistics from

[1]See `http://www.itl.nist.gov/iaui/894.02/related_projects/muc` for details on MUC.

[2]See `http://www.itl.nist.gov/iad/894.01/tests/ace` for details on the ACE research program.

[3]For instance, the minimum number of coreference links needed to put n NPs in the same coreference equivalence class is n-1. We are interested in the minimum number of links because this is what our coreference scoring program is concerned

Table 5.1: Statistics of the MUC and ACE data sets.

	MUC		ACE		
	MUC-6	MUC-7	BNEWS	NPAPER	NWIRE
Number of texts	60	50	267	93	159
Training	30	30	216	76	130
Testing	30	20	51	17	29
Number of tokens	29188	28043	85827	90188	106216
Training	13919	16700	67470	71944	85688
Testing	15199	11343	18357	18174	20528
Number of markables	4232	4297	9316	10674	10076
Training	2141	2569	7291	8749	8183
Testing	2091	1728	2025	1925	1893
Number of coref links	3169	3179	7153	8781	7987
Training	1592	1887	5596	7221	6505
Testing	1577	1292	1557	1560	1482

the training and testing portions of each data set.

Although both the MUC and ACE corpora are standard corpora used in coreference evaluation, ACE appears to be a better choice from an evaluation perspective. As we can see from the table, the ACE data sets are larger than the MUC data sets in terms of both the number of texts and the number of tokens. Perhaps more importantly, ACE provides much more labeled data for training: a substantially larger number of *positive* instances can be generated from the training texts due to the presence of a large number of coreference links. Furthermore, ACE is more with, as we will see in Section 5.2.

"diverse" than its MUC counterpart with respect to the *type* of news sources: while the MUC data sets are composed exclusively of newswire articles, ACE consists of articles drawn from broadcast news and newspapers in addition to newswire, as mentioned above.

Given that coreference is a fairly complex linguistic phenomenon, it is not surprising that humans may disagree on what constitutes a coreference relationship between two noun phrases. To standardize the notion of which pair of NPs should be marked up as co-referring, both MUC and ACE have their own set of guidelines. Not only do the human annotators use these guidelines when marking up coreference chains, the MUC and ACE participants are expected to follow them. Note that we do not have to worry about the specifics of these guidelines: since these guidelines are used to create the annotated corpora from which our coreference rules are learned, they should in principle be automatically reflected in the rules.

5.2 Evaluation Metric

Both MUC and ACE have their own scoring program for evaluating the output of a coreference system. Throughout this chapter, however, we will evaluate the system using the model-theoretic MUC scoring program (Vilain et al., 1995) for both the MUC and ACE data sets. This is primarily because (1) our evaluation framework is set up to perform MUC-style evaluation and (2) a considerable amount of engineering effort is needed to transform the input assumed by the MUC scorer to that assumed by the ACE scorer. We are not particularly concerned about not using the ACE scoring program when evaluating the system on the ACE corpus, in part because we would not be able to directly compare our results with other published ACE results such as McCallum and Wellner (2003) and Luo et al. (2004) even if

we used the ACE scorer. Specifically, these ACE coreference systems rely on the pre-extracted NP markables provided by the ACE organizers. On the other hand, our system extracts the markables from the input texts automatically, since it is set up to perform MUC-style evaluation, as mentioned above.

The MUC scoring program reports system performance in terms of three metrics commonly used in the information retrieval community: *recall*, *precision*, and *F-measure*. Intuitively, to compute recall and precision for a given a linguistic task, we compare a set of linguistic "objects" we want our system to identify (A) with a set of objects that the system actually identifies (B). Assuming that C is the intersection of A and B, recall is equal to $\frac{|C|}{|A|}$ (i.e., the fraction of the objects in A identified by the system), and precision is equal to $\frac{|C|}{|B|}$ (i.e., the fraction of the objects in B that are correctly identified). Hence, recall and precision can informally be viewed as a measure of *task coverage* and *task accuracy*, respectively. For evaluation purposes, recall and precision are usually combined into a single metric known as F-measure:

$$\text{F-measure} = \frac{(\beta^2 + 1) \times \text{recall} \times \text{precision}}{\text{recall} + \beta^2 \times \text{precision}},$$

where β is a free parameter that allows the evaluator to adjust the relative importance of recall and precision. Following standard practice, we set β to one in our experiments, placing equal weight on recall and precision.

For coreference resolution, it seems natural to define the linguistic "objects" we want to identify to be the coreference links. Nevertheless, the MUC scoring program does not attempt to compare the coreference links in the key partition (the correct partition of the NPs) with those in the response partition (the partition generated by a coreference system). Rather, the program compares the clusters (or equivalence classes) as defined by the links in the two partitions. In other

words, the scoring program abstracts away from which coreference links are used to generate a cluster in a given partition.[4]

Computing recall. To define recall based on cluster comparison rather than link comparison, we will make use of the following notations.

- S_i is the i-th cluster in the key partition S.
- R_1, R_2, ..., R_m is the set of clusters in the response partition R.
- $p(S_i)$ is a partition of S_i relative to the response. It should be easy to see that if all the objects in S_i are in the same cluster in the response partition, then $|p(S_i)| = 1$; on the other hand, if the objects in S_i are in $|S_i|$ different clusters in the response partition, then $|p(S_i)| = |S_i|$.
- $c(S_i)$ is the minimal number of correct links needed to create S_i. It should be clear that $c(S_i) = |S_i|$ - 1.
- $m(S_i)$ is the number of missing links in the response relative to the key set S_i. It should be clear that $m(S_i) = |p(S_i)|$ - 1.

Using the above notations, we can define recall as follows:

$$\text{recall} = \frac{\sum_i c(S_i) - m(S_i)}{\sum_i c(S_i)} \tag{5.1}$$

$$= \frac{\sum_i (|S_i| - 1) - (|p(S_i)| - 1)}{\sum_i |S_i| - 1} \tag{5.2}$$

$$= \frac{\sum_i |S_i| - |p(S_i)|}{\sum_i |S_i| - 1} \tag{5.3}$$

[4]To see the difference between comparing links and comparing clusters, simply take note of the fact that there are numerous ways to create a cluster consisting of n objects. For instance, discovering *any* n-1 links among the objects will do.

Computing precision. Precision can be similarly computed by reversing the role of the key partition and the response partition in the definition of recall. More specifically, we will make use of the following notations in defining precision.

- R_i is the i-th cluster in the response partition R.
- S_1, S_2, ..., S_n is the set of clusters in the key partition S.
- $p(R_i)$ is a partition of R_i relative to the key. It should be easy to see that if all the objects in R_i are in the same cluster in the key partition, then $|p(R_i)| = 1$; on the other hand, if the objects in R_i are in $|R_i|$ different clusters in the key partition, then $|p(R_i)| = |R_i|$.
- $c(R_i)$ is the minimal number of correct links needed to create R_i. It should be clear that $c(R_i) = |R_i|$ - 1.
- $m(R_i)$ is the number of missing links in the key relative to the response set R_i. It should be clear that $m(R_i) = |p(R_i)|$ - 1.

Using the above notations, we can define precision as follows:

$$\text{precision} = \frac{\sum_i c(R_i) - m(R_i)}{\sum_i c(R_i)} \tag{5.4}$$

$$= \frac{\sum_i (|R_i| - 1) - (|p(R_i)| - 1)}{\sum_i |R_i| - 1} \tag{5.5}$$

$$= \frac{\sum_i |R_i| - |p(R_i)|}{\sum_i |R_i| - 1} \tag{5.6}$$

Now that we know how to compute recall and precision for a given document, we can extend the method to compute recall and precision for a given collection of documents.

Computing overall recall. Assume that S_i^k is the i-th cluster in the key partition S^k for document Doc_k. By (5.3), we have

$$\text{recall}(Doc_k) \quad = \quad \frac{\sum_i |S_i^k| - |p(S_i^k)|}{\sum_i |S_i^k| - 1} \tag{5.7}$$

Now, we can define the recall fraction for a collection of documents, $Coll$, as follows:

$$\text{recall}(Coll) \quad = \quad \frac{\sum_{Doc_k \in Coll} \sum_i |S_i^k| - |p(S_i^k)|}{\sum_{Doc_k \in Coll} \sum_i |S_i^k| - 1} \tag{5.8}$$

Comparing (5.7) and (5.8), we can see that the numerator of recall($Coll$) is computed by summing the numerators of the recall fractions of the documents in $Coll$. Similarly, its denominator is computed by summing the denominators of the recall fractions of the documents in $Coll$.

Computing overall precision. Overall precision can be similarly computed. Assume that R_i^k is the i-th cluster in the response partition R^k for document Doc_k. By (5.6), we have

$$\text{precision}(Doc_k) \quad = \quad \frac{\sum_i |R_i^k| - |p(R_i^k)|}{\sum_i |R_i^k| - 1} \tag{5.9}$$

Now, we can define the precision fraction for a collection of documents, $Coll$, as follows:

$$\text{precision}(Coll) \quad = \quad \frac{\sum_{Doc_k \in Coll} \sum_i |R_i^k| - |p(R_i^k)|}{\sum_{Doc_k \in Coll} \sum_i |R_i^k| - 1} \tag{5.10}$$

Comparing (5.9) and (5.10), we can see that the numerator of precision($Coll$) is computed by summing the numerators of the precision fractions of the documents in $Coll$. Similarly, its denominator is computed by summing the denominators of the precision fractions of the documents in $Coll$.

Computing overall F-measure. The F-measure achieved by a coreference system on *Coll* is simply the balanced harmonic mean of overall recall and precision. Note that this scoring method essentially places more importance on documents with a larger number of coreference links than those with a smaller number of links.

Potential problems with the MUC scoring program. Note from the above discussion that the MUC scoring program rewards the correctly identified coreference links by increasing recall and penalizes the spuriously identified ones by decreasing precision. However, the scorer does not directly reward successful identification of *non-coreference* relationships, and has been criticized for its asymmetric treatment of coreference and non-coreference (Bagga and Baldwin, 1998). Despite its potential weakness, using the MUC scoring program for evaluation purposes facilitates performance comparison between our system and existing systems.

5.3 Data Pre-processing

When we discussed the **GenericResolve** algorithm in Chapter 2, we pointed out that identification of discourse entities is the first step taken in virtually all existing anaphora and coreference resolution systems. Indeed, it is the main task in preprocessing an input text to be analyzed by a coreference system. In principle, we can either use a partial parser to extract the base, non-recursive noun phrases from a text, or a full parser to recover all noun phrases from the corresponding parse trees. In practice, neither of them is entirely satisfactory. Considering only base noun phrases in coreference analysis may severely limit the recall of the coreference system, since many coreference relationships are formed with higher-level NPs that

are structurally more complicated than base NPs. On the other hand, although using all NPs extracted from parse trees can potentially avoid the aforementioned problem with recall, this method creates another problem: different NPs may be textually realized by the same underlying NP. To see what this means, let us consider the following sentence.

> *King George VI of Great Britain, the husband of Queen Elizabeth, died at the age of 56.*

It should be clear that although the three NPs *King George VI*, *King George VI of Great Britain*, and *King George VI of Great Britain, the husband of Queen Elizabeth* would appear as distinct NPs in the parse tree of this sentence, they are textually realized by the same underlying NP. Hence, if we use all three NPs in our coreference analysis, the system can potentially create "repetitive" coreference links, thus damaging its precision.

In an attempt to address both of the problems mentioned above, we employ the procedure **ExtractMarkables** shown in Figure 5.1 to extract NP markables from a text. The basic idea behind **ExtractMarkables** is to augment the set of discourse entities extracted by a base NP finder with the help of a named entity finder. Specifically, the algorithm uses a variable *ExtractedNPs* to store the set of discourse entities extracted from the given text thus far. The variable is initialized to the set of base NPs extracted from the given text via an in-house base noun phrase finder (Cardie and Pierce, 1998). Then, each base NP is compared to a set of named entities extracted from the text by a statistical named entity finder (Bikel et al., 1999). If the base NP partially overlaps with a named entity, then it is expanded so that its text span subsumes that of the named entity. On the other hand, if its text span already subsumes that of a named entity, the named entity

---

ExtractMarkables(D)

Input: A document D

Algorithm:
$ExtractedNPs$:= the set of NPs extracted from D by a base NP finder
NEs := the set of named entities extracted from D by a named entity finder

foreach noun phrase np in $ExtractedNPs$ **do**
  if np partially overlaps with a named entity ne in NEs
    Expand np so that its text span subsumes that of ne.
  else if there exists an ne in NEs such that ne is a proper substring of np
    Add ne to $ExtractedNPs$.
endfor

Add to $ExtractedNPs$ all named entities in NEs that do not overlap with any element in $ExtractedNPs$.

Add to $ExtractedNPs$ all nested nouns (including possessive pronouns) within a base NP that is not a named entity.

Optionally prune the elements in $ExtractedNPs$ according to corpus-specific annotation guidelines.

Output: $ExtractedNPs$

Figure 5.1: The **ExtractMarkables** Algorithm

is added to $ExtractedNPs$. After that, all the named entities that do not overlap with any of the elements in $ExtractedNPs$ are added to $ExtractedNPs$. Finally, the algorithm adds all nested nouns or NPs within a base NP that is not a named entity. Roughly speaking, there are two types of nested nouns or NPs. The first type includes possessive noun phrases (e.g., *my neighbor* in the noun phrase *my neighbor's dog*) and possessive pronouns. The second type includes modifiers to head nouns or NPs (e.g., *wage* in the noun phrase *wage reductions*).

At the end of the algorithm, there is an optional step for pruning the markables based on corpus-specific annotation guidelines. For the MUC-6 data set, NPs that lie outside the text delimited by the TXT, HL, DD, and DATELINE tags as well as those that appear within a table are discarded.[5] For the MUC-7 data set, NPs that lie outside the text delimited by the SLUG, DATE, NWORDS, PREAMBLE, TEXT, and TRAILER tags are discarded.[6] For the ACE data sets, preliminary experiments with the training data indicate that it is more beneficial to discard nested NPs that are common nouns.

To ensure that **ExtractMarkables** is a reasonably effective procedure for extracting markables from a text, we conduct the **All One Cluster** experiment on the training texts of each of the five data sets. In **All One Cluster**, we generate for each training text a dummy partition in which all NPs are put into the same cluster and use the MUC scoring program to compute the overall recall score. This experiment is useful in that it establishes an upper bound on recall for any coreference system using this set of NPs for analysis.[7] In other words, the recall score provides an indirect indication of how well the NP extraction procedure works. The recall scores for the five data sets are 93.8 (MUC-6), 90.2 (MUC-7), 96.7 (BNEWS), 96.0 (NPAPER), and 88.5 (NWIRE). These results suggest that **ExtractMarkables** is a fairly effective procedure for extracting markables for a coreference system.

[5]See the MUC-6 coreference task definition (`http://www.cs.nyu.edu/cs/faculty/grishman/COtask21.book_1.html`) for a more detailed description of which part of a text a system is expected to annotate.

[6]See the MUC-7 coreference task definition (`http://www.itl.nist.gov/iaui/894.02/related_projects/muc/proceedings/co_task.html`) for a more detailed description of which part of a text a system is expected to annotate.

[7]Note that the upper bound on recall is 100% if and only if the set of NPs identified by the coreference system is a superset of the set of NPs involved in some coreference relationship according to the answer key.

5.4 Results

After the markables have been identified from a text using **ExtractMarkables**, we can present them to the coreference system for analysis. In this section, we will evaluate our coreference system on the MUC and ACE data sets using the MUC scoring program. Below we will first describe and evaluate a set of baseline coreference algorithms. The first four are previously proposed by Cardie and Wagstaff (1999) and the last one is our duplication of the Soon et al. (2001) system.

5.4.1 Baseline Systems

Baseline 1: All One Cluster. Recall from the preceding section that the **All One Cluster** baseline naively assumes that all NPs presented to the coreference system for analysis are coreferent. In other words, this baseline system always creates a dummy partition in which all NPs are in the same cluster for each text it processes. As mentioned before, **All One Cluster** establishes an upper bound on recall for a coreference system using the given set of NPs for analysis. Results using this baseline for each of the five data sets are shown in row 1 of Tables 5.2-5.6, where performance is reported in terms of recall, precision, and F-measure. As we can see, **All One Cluster** establishes an F-measure of 49.2 (MUC-6), 48.9 (MUC-7), 40.2 (BNEWS), 43.6 (NPAPER), and 31.8 (NWIRE).

Baseline 2: Any Word Match. The choice of our second baseline, **Any Word Match**, is motivated by the observation that two NPs that share a common word are more likely to be coreferent than those that do not. Specifically, **Any Word Match** posits a coreference relationship between any two NPs that have a word in common. Results using this baseline are shown in row 2 of Tables 5.2-5.6. As

Table 5.2: Results for the MUC-6 data set. Boldface indicates the best results.

	System	Recall	Precision	F-measure
1	All One Cluster	93.8	33.4	49.2
2	Any Word Match	62.7	45.1	52.5
3	Head Noun Match	58.8	53.9	56.2
4	Exact String Match	46.1	73.4	56.6
5	Duplicated Soon	61.8	70.1	65.7
6	Our System	75.8	61.4	**67.9**
7	Pronouns only	24.9	67.3	36.3
8	Proper Nouns only	36.9	82.8	51.1
9	Common Nouns only	30.7	47.5	37.3

we can see, **Any Word Match** achieves an F-measure of 52.5 (MUC-6), 49.2 (MUC-7), 46.3 (BNEWS), 47.2 (NPAPER), and 36.4 (NWIRE). Since, unlike **All One Cluster**, this baseline does not blindly put all NPs into the same cluster, we see gains in precision accompanied by loss in recall.

Baseline 3: Head Noun Match. Our third baseline, **Head Noun Match**, tightens the constraint imposed by **Any Word Match** on when two NPs can be coreferent. Specifically, **Head Noun Match** marks two NPs as coreferent if and only if their head nouns match. It should be easy to see that any two NPs that are marked as coreferent by **Head Noun Match** are also considered coreferent according to the **Any Word Match** criterion. Consequently, we hypothesized that recall would drop in comparison to Baseline 2. Results using this baseline are shown in row 3 of Tables 5.2-5.6. As we can see, **Head Noun Match** achieves

Table 5.3: Results for the MUC-7 data set. Boldface indicates the best results.

	System	Recall	Precision	F-measure
1	All One Cluster	90.7	33.5	48.9
2	Any Word Match	58.4	42.5	49.2
3	Head Noun Match	54.8	53.7	54.3
4	Exact String Match	40.7	72.4	52.1
5	Duplicated Soon	55.2	68.3	61.1
6	Our System	64.2	60.2	**62.1**
7	Pronouns only	11.2	54.4	18.6
8	Proper Nouns only	28.3	75.7	41.2
9	Common Nouns only	28.6	52.3	37.0

Table 5.4: Results for the BNEWS data set. Boldface indicates the best results.

	System	Recall	Precision	F-measure
1	All One Cluster	93.8	25.6	40.2
2	Any Word Match	57.6.	38.7	46.3
3	Head Noun Match	52.8	39.9	45.4
4	Exact String Match	45.9	52.3	48.9
5	Duplicated Soon	53.7	47.6	50.5
6	Our System	63.1	67.8	**65.4**
7	Pronouns only	35.8	68.6	47.0
8	Proper Nouns only	39.0	68.8	49.8
9	Common Nouns only	7.1	63.2	12.7

Table 5.5: Results for the NPAPER data set. Boldface indicates the best results

	System	Recall	Precision	F-measure
1	All One Cluster	95.3	28.3	43.6
2	Any Word Match	59.4	39.2	47.2
3	Head Noun Match	56.7	41.2	47.7
4	Exact String Match	47.6	57.1	51.9
5	Duplicated Soon	63.3	56.7	59.8
6	Our System	73.5	63.3	**68.0**
7	Pronouns only	50.1	63.0	55.8
8	Proper Nouns only	48.8	67.9	56.8
9	Common Nouns only	9.4	49.3	15.7

Table 5.6: Results for the NWIRE data set. Boldface indicates the best results

	System	Recall	Precision	F-measure
1	All One Cluster	85.3	19.6	31.8
2	Any Word Match	52.2	27.9	36.4
3	Head Noun Match	47.3	30.9	37.4
4	Exact String Match	38.9	42.8	40.8
5	Duplicated Soon	49.3	41.1	44.8
6	Our System	53.1	60.6	**56.6**
7	Pronouns only	24.0	59.3	34.1
8	Proper Nouns only	37.6	66.8	48.1
9	Common Nouns only	7.0	48.6	12.3

an F-measure of 56.2 (MUC-6), 54.3 (MUC-7), 45.4 (BNEWS), 47.7 (NPAPER), and 37.4 (NWIRE). These results are consistent with our hypothesis: recall drops in all five data sets.

Baseline 4: Exact String Match. Our fourth baseline, **Exact String Match**, further strengthens the constraint imposed by **Head Noun Match** on when two NPs can be coreferent. Specifically, **Exact String Match** marks two NPs as coreferent if and only if the strings denoting the two NPs are identical. Since this coreference constraint is more stringent than those in **Head Noun Match** and **Any Word Match**, we hypothesized that recall would drop in comparison to these baselines. Results using **Exact String Match** are shown in row 4 of Tables 5.2-5.6. As we can see, performing exact string match enables us to achieve an F-measure of 56.6 (MUC-6), 52.1 (MUC-7), 48.9 (BNEWS), 51.9 (NPAPER), and 40.8 (NWIRE). Again, these results are consistent with our hypothesis: recall drops in all five data sets.

Baseline 5: Duplicated Soon. The choice of our last baseline, **Duplicated Soon**, is motivated by the fact that the Soon et al. (2001) system is the first learning-based coreference engine that achieves performance comparable to the best MUC coreference systems. **Duplicated Soon** attempts to duplicate both the approach and the knowledge sources employed in Soon et al. (2001). A detailed description of the Soon et al. system can be found in Section 3.3.2. Basically, their system can be viewed as an instantiation of our parametric approach to coreference, employing a decision tree learner that has access to 12 surface-level features and the closest-first clustering algorithm to coordinate the coreference decisions made by the coreference classifier.

Table 5.7: Performance of the original Soon system and our Duplicated Soon baseline on the MUC data sets

	MUC-6			MUC-7		
System Variation	Recall	Precision	F	Recall	Precision	F
Original Soon et al.	58.6	67.3	62.6	56.1	65.5	60.4
Duplicated Soon	61.8	70.1	65.7	55.2	68.3	61.1

Results using the **Duplicated Soon** baseline are shown in row 5 of Tables 5.2-5.6. As we can see, this baseline achieves an F-measure of 67.9 (MUC-6), 61.1 (MUC-7), 50.5 (BNEWS), 59.8 (NPAPER), and 44.8 (NWIRE). This translates to an improvement of 9% in F-measure for the MUC data sets. Somewhat surprisingly, the performance improvement is less substantial for the ACE data sets (in particular for BNEWS and NWIRE) despite the availability of a larger amount of training data provided by ACE. This implies that either the test cases in the ACE data are harder to resolve or the Soon approach is unable to exploit the potential benefits from having additional training data. Interestingly, while the MUC results suggest that **Duplicated Soon** is a high-precision/low-recall system, the ACE results seem to characterize the system as having low precision/high recall.

To ensure that our re-implementation of the Soon et al. system is a reasonable duplication, we compare our results with those that Soon et al. (2001) reported. For both data sets, our results are at least as strong as the original Soon results (see Table 5.7). The primary reason for improvements over the original Soon system for the MUC-6 data set appears to be our higher upper bound on recall (93.8% vs. 89.9%), due to better identification of NPs. For MUC-7, our improvement stems from increases in precision, presumably due to more accurate feature value

computation. Overall, these results indicate that our Soon re-implementation is a reasonable duplication of Soon et al.'s original system.

Comparison with the MUC coreference systems. To get a more concrete idea of how "strong" these baselines are, we compare the performance of our baselines with that of the MUC coreference systems. Figures 5.2 and 5.3 show the performance of these systems (sorted by F-measure scores) on the MUC-6 and MUC-7 test data. As we can see, there are seven participating systems in both MUCs.[8] For each data set, our five baselines strictly outperform two MUC systems. In particular, our **Duplicated Soon** baseline outperforms all of the MUC systems on the MUC-6 test data and achieves performance comparable to the best system on the MUC-7 test data. These results provide suggestive evidence that our baseline systems are reasonably strong.

5.4.2 Our Coreference System

Recall from Section 4.7 that our parameter computation algorithm, **TuneCorefParams**, relies on held-out data to compute the parameters of our parametric coreference system. To ensure a fair comparison between our approach and other baselines, we do not rely on additional labeled data for parameter tuning in the former. Instead, as discussed in Section 4.7, **TuneCorefParams** uses $\frac{2}{3}$ of the available training texts for learning coreference and anaphoricity classifiers and reserving the rest for development.[9]

[8]It should be noted that all of the MUC coreference systems are knowledge-based, with the exception of University of Massachusetts' coreference engine.

[9]In all of our experiments, we simply use the first $\frac{2}{3}$ of the texts in the training corpus for acquiring coreference and anaphoricity classifiers. In particular, we do not perform any cross-validation.

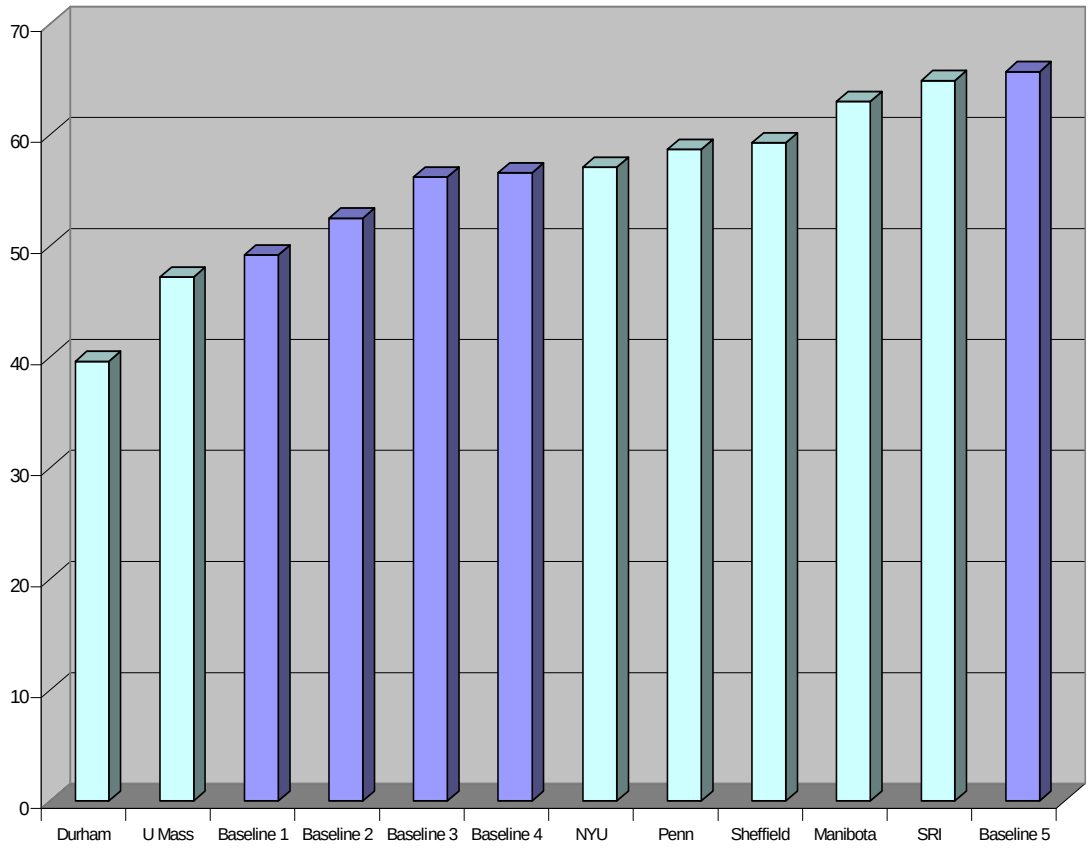

Figure 5.2: F-measure scores of the MUC coreference systems and our baselines on the MUC-6 test data

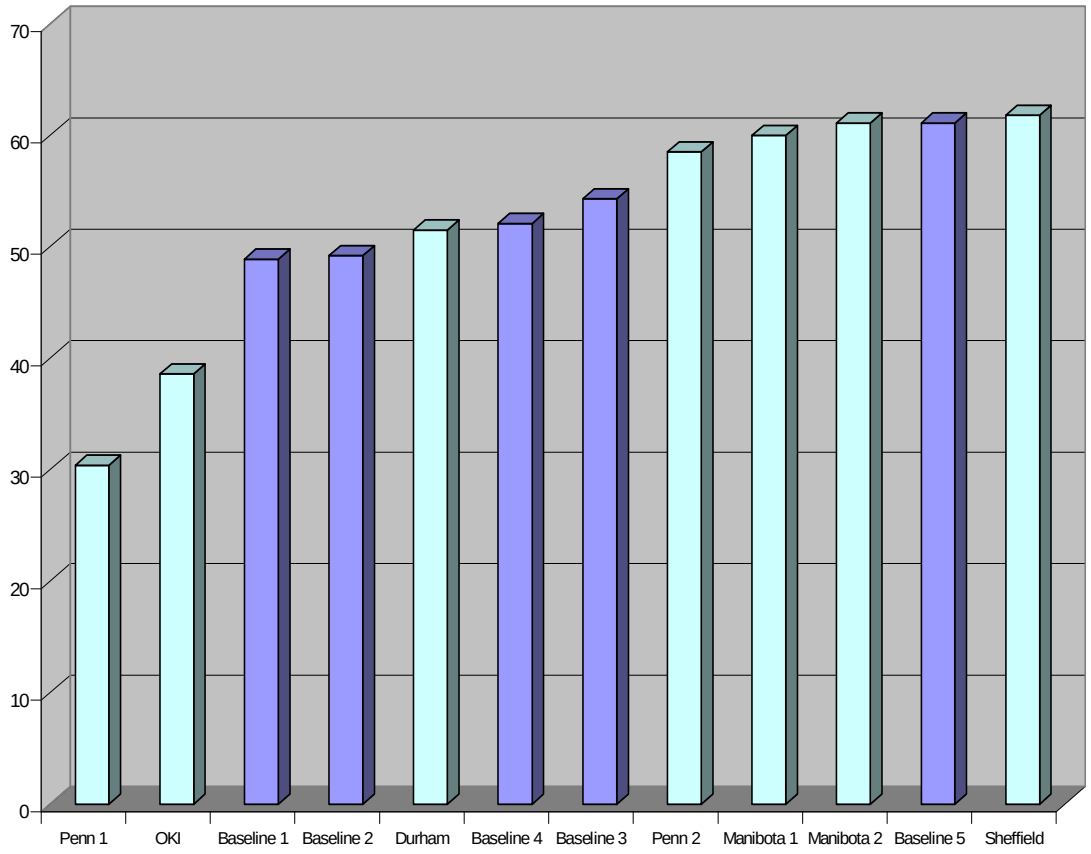

Figure 5.3: F-measure scores of the MUC coreference system and our baselines on the MUC-7 test data

Table 5.8: Coreference system configurations that achieve the best performance on held-out development data

Data set	C	I	R	L	F	A
MUC-6	aggressive	Soon-2	yes	RIPPER	expanded	cr=6
MUC-7	aggressive	Soon-1	no	C4.5	expanded	cr=6
BNEWS	aggressive	M&L	no	C4.5	expanded	cr=8
NPAPER	aggressive	M&L	no	C4.5	expanded	cr=3
NWIRE	aggressive	M&L	no	C4.5	expanded	cr=4

The best coreference system configurations determined by this method are shown in Table 5.8, where each row represents the best configuration learned for the associated data set. Recall that under our parameterization, each coreference system is represented as a 6-tuple comprising a *clustering algorithm* (C), a *training instance creation method* (I), a decision of whether to apply *rule pruning* (R), a *learning algorithm* (L), a *coreference feature set* (F), and a cost ratio specifying how to train an *anaphoricity classifier* (A). Now, two points deserve mentioning.

- Our empirically-determined coreference systems are quite different from Soon et al.'s system. First, all of them make use of aggressive-merge clustering as opposed to Soon et al.'s closest-first clustering. Second, the consistent preference for the expanded feature set over the Soon feature set provides indirect evidence that incorporating additional knowledge can improve system performance. Finally, anaphoricity determination appears to be a useful component to all of the best-performing systems, although the cost ratio employed differs among the systems.

- The best configurations for the three ACE data sets are basically the same except for the cost ratio used to train the anaphoricity classifier. In particular, all of them are constructed by acquiring a decision tree classifier on the training instances created via McCarthy and Lehnert's method, where each instance is represented by our expanded feature set. The major difference between the best ACE systems and the best MUC systems lies in the training instance creation scheme employed: while the ACE systems favor the McCarthy and Lehnert method (in which training instances are created from all distinct NP pairs in a training text), the MUC systems prefer the Soon-n method with small values of n. This contrasts with the general belief that learning from highly skewed data sets is always a problem for coreference systems. It is possible that preserving information in the training instances is more crucial than reducing data skewness for the ACE systems, whereas the reverse is true for the MUC systems. Table 5.9 provides further details on the size and skewness of the data sets on which the coreference classifiers underlying the best-performing systems are trained.

Results on the test data. Now that we have the best coreference system configurations, we can evaluate them on the test data. Results of our coreference systems are shown in row 6 of Tables 5.2-5.6. As we can see, our system outperforms the best baseline — **Duplicated Soon** — on all five data sets, achieving an F-measure of 67.9 (MUC-6), 62.1 (MUC-7), 65.4 (BNEWS), 68.0 (NPAPER), and 56.6 (NWIRE).[10] Performance improvements (in terms of F-measure) are partic-

[10]An earlier version of our system (see Ng and Cardie (2002c)) achieves even better MUC results (F-measure of 70.4 and 63.4 for the MUC-6 and MUC-7 data sets, respectively) than those reported here. As described in that paper, the better

Table 5.9: The size and skewness of the data sets on which the classifiers underlying the best-performing coreference systems are trained

Data set	Number of training insts	Number of positive insts	% positives
MUC-6	26588	6343	23.86
MUC-7	22599	4271	18.90
BNEWS	1046763	22948	2.19
NPAPER	2290609	46491	2.03
NWIRE	1729881	14504	0.84

ularly pronounced on the ACE data sets: F-measure increases by approximately 15%, 8%, and 12% for the BNEWS, NPAPER, and NWIRE data sets, respectively, as opposed to only 1-2% on the MUC data sets.

There are two plausible reasons for the relatively small performance improvement over **Duplicated Soon** for the two MUC data sets. First, the Soon system configuration (i.e., a decision tree learner with 12 features and the closest-first algorithm) may simply work well on the MUC data. Hence, the observed benefit from empirically determining the best system configuration using our approach may be marginal. Second, the amount of labeled data available for training in MUC (see Table 5.1) may be inadequate for our approach to work well. Recall that, unlike

results are obtained by training the coreference classifier on a set of features manually selected from our expanded feature set. We have considered using this feature set in our current work, but are concerned by the fact that a fair amount of user discretion was employed to discard features used to induce low-precision rules during the manual feature selection process. As mentioned in Section 4.7, we attempted to automate the feature selection process by employing a forward selection algorithm and a backward elimination algorithm, but the process is computationally very expensive and yet is not effective in improving system performance. Hence, we neither employ the manually selected features nor perform automatic feature selection in our current work.

Soon et al., our system relies on the labeled data not only for training coreference classifiers but also for parameter computation. Therefore, the reduced amount of labeled data used for training coreference classifiers (in comparison to Soon) may adversely affect classifier performance, and the small amount of labeled data we set aside for development in MUC may be insufficient for determining a good set of parameters.

Perhaps more importantly, **Duplicated Soon** performs fairly poorly on the ACE data sets, as mentioned previously. This offers indirect evidence for the claim that the Soon system configuration may not be universally good. (Note that the Soon approach is originally developed for and has only been tested on the MUC data sets.) On the other hand, our approach seems to work quite well across the data sets, although it is clear that there is substantial room for improvements. The parameterization employed by **TuneCorefParams** may be responsible for the superior performance of our approach: it may have allowed us to better capture the specificities of each data set as well as the complex interactions among different components of the coreference system.

User-defined objective function and system selection criterion. It is interesting to note that in the MUC-6, MUC-7, and NPAPER data sets, our coreference system achieves higher recall than precision — a characteristic that distinguishes itself from all of the MUC coreference engines, in which precision is always higher than recall. The high recall levels achieved on these data sets can be attributed in part to the use of the aggressive-merge clustering algorithm, which, as discussed previously, encourages more merging than other clustering algorithms such as best-first and closest-first.

In applications where precision is critical, these empirically determined "high-recall" systems may be less preferable to their precision-oriented counterparts. However, this is by no means a limitation of our approach. If we desire high-precision systems, we can simply alter the objective function employed by **TuneCorefParams** (which is currently F-measure with β set to one) to put extra emphasis on precision. To see how this can be done, re-consider the definition of F-measure:

$$\text{F-measure} = \frac{(\beta^2 + 1) \times \text{recall} \times \text{precision}}{\text{recall} + \beta^2 \times \text{precision}},$$

where β is a free parameter to the function. As mentioned previously, β allows us to adjust the relative importance of recall and precision: decreasing (increasing) β places more (less) weight on precision than recall. In general, our approach is flexible enough to allow users to define their (application-dependent) objective function employed by **TuneCorefParams** and accommodate their preferences in the system selection criterion.

Performance on specific NP types. The performance of the system on different NP types including pronouns, proper nouns, and common nouns is shown in rows 7-9 of Tables 5.2-5.6.[11] We obtain the numbers for a given NP type (say pronouns) by having the system attempt to resolve only the pronouns in the test texts. So the system response will contain only coreference chains generated by resolving the pronouns. However, the answer key that we provide the scoring program for computing these numbers is always the one that contains coreference chains for *all* types of NPs.

[11]Here, we treat an NP as a common noun if it is neither a pronoun nor a proper noun.

In accordance with our intuition, the results of our system also show that among the different NP types, proper noun resolution achieves the highest precision, whereas common noun resolution achieves the lowest (see rows 7-9 of Tables 5.2-5.6). This trend is more pronounced for the MUC data sets than for the ACE data sets. Nevertheless, care should be taken in interpreting these results, since the precision scores achieved by a system are dependent in part on the underlying clustering algorithm employed. For instance, had the system employed a precision-oriented clustering algorithm (e.g., best-first clustering), the precision scores might have increased.

5.5 Chapter Summary

In this chapter, we evaluated our machine learning approach to coreference resolution on five standard data sets derived from the MUC and ACE coreference corpora. Our experiments demonstrated the consistently superior performance of our approach to the best existing learning-based coreference system — the Soon et al. system — and provided reasonably convincing evidence that our ultimate goal — improving machine learning approaches to coreference resolution — has been accomplished.

CHAPTER 6

PERFORMANCE ANALYSIS

> *We cannot trust the results of a quantitative evaluation without doing a considerable amount of qualitative analyses and we should perform our qualitative analyses on those components that make a significant contribution to the quantitative results; we need to be able to measure the effect of various factors.* — Walker (1989)

In the previous chapter, we showed that our coreference systems outperform **Duplicated Soon**, our re-implementation of the best existing learning-based coreference system, on five standard coreference data sets. In particular, they offer reasonably convincing evidence that our goal — improving existing machine learning approaches to coreference resolution — has been accomplished. To better understand the strengths and weaknesses of our approach to coreference resolution, we will analyze its performance in this chapter.

6.1 Feature and Classifier Analyses

We will first focus on linguistic issues pertaining to coreference resolution and anaphoricity determination in our analysis. Specifically, we examine in this section what *rules* and *features* are important to coreference resolution and anaphoricity determination, with the goal of gaining additional insights into the linguistic aspects of these problems.

```
+ 740 100 IF WORD_OVERLAP = C .
+ 103 12 IF WNCLASS = C PRO_RESOLVE = C ANTE_MED >= 0.0645 .
+ 7 0 IF WNCLASS = C CONSTRAINTS = NA SENTNUM <= 1 ANTE_MED >= 0.0555
   SUBJECT_1 = Y ANTE_MED <= 0.0909 .
+ 21 8 IF WNCLASS = C SENTNUM <= 2 BOTH_PROPER_NOUNS = NA NUMBER = C
   BOTH_SUBJECTS = C ANA_MED <= -0.1 .
+ 13 0 IF WNCLASS = C CONSTRAINTS = NA SENTNUM <= 1 APPOSITIVE = C .
+ 11 1 IF WNCLASS = C CONSTRAINTS = NA SENTNUM <= 1 ANTE_MED >= 0.5
   ANIMACY = C .
+ 36 10 IF WNCLASS = C PROPER_NOUN = C BOTH_PROPER_NOUNS = C
   NUMBER = C .
+ 14 6 IF WNCLASS = C CONSTRAINTS = NA PARANUM <= 1 PARANUM <= 0
   GRAMROLE_2 = gen BOTH_SUBJECTS = NA .
+ 7 0 IF WNCLASS = C CONSTRAINTS = NA PRONOUN_2 = Y ANTE_MED >= 0.6 .
+ 13 6 IF WNCLASS = C BOTH_PROPER_NOUNS = NA SENTNUM <= 3 SUBCLASS = C
   AGREEMENT = C DEFINITE_2 = Y ANA_MED <= 0.1818 GRAMROLE_1 = null .
+ 20 13 IF WNCLASS = C PROPER_NOUN = C NUMBER = C SENTNUM <= 3
   BOTH_PRONOUNS = NA AGREEMENT = C SUBJECT_2 = Y ANTE_MED <= -1 .
+ 8 0 IF WNCLASS = C BOTH_PROPER_NOUNS = NA NUMBER = C SENTNUM <= 3
   BOTH_DEFINITES = NA ANTE_MED <= -0.25 ANTE_MED >= -0.3333 .
- 24731 597 IF .
.
```

Figure 6.1: Best-performing classifier for the MUC-6 development data

6.1.1 Analyzing Coreference Rules and Features

Coreference rules. To get an idea of the rules that are useful for coreference resolution, we show the coreference classifiers underlying the best-performing coreference systems on the MUC-6 and NPAPER development data in Figures 6.1 and 6.2. As we can see, the MUC-6 classifier is a propositional ruleset, whereas the NPAPER classifier is a decision tree.[1]

In the MUC-6 classifier, each rule induced by RIPPER is composed of the classification value (+ [COREFERENT] or − [NOT COREFERENT]) and a condition (the conjunction of attribute-value pairs after IF) specifying when the rule is applicable. In addition, the two integers immediately following the classification value indicate the number of training instances correctly and incorrectly covered by the

[1]Note that we only show the top five levels of the decision tree, since the whole tree prints on more than 50 pages.

```
PN_STR = C:
|   SUBCLASS = I: + (8549.0/765.8)
|   SUBCLASS = C:
|   |   SUBJECT_1 = Y: + (3.0/1.1)
|   |   SUBJECT_1 = N:
|   |   |   SUBJECT_2 = Y: + (2.0/1.0)
|   |   |   SUBJECT_2 = N: - (18.0/3.7)
PN_STR = I:
|   PRO_EQUIV = C:
|   |   WORD_OVERLAP = C:
|   |   |   BOTH_SUBJECTS = NA: - (721.0/50.2)
|   |   |   BOTH_SUBJECTS = I: - (5.0/1.2)
|   |   |   BOTH_SUBJECTS = C:
|   |   |   |   AGREEMENT = NA: + (9.0/1.3)
|   |   |   |   AGREEMENT = I: - (0.0)
|   |   |   |   AGREEMENT = C: - (37.0/9.3)
|   |   WORD_OVERLAP = I:
|   |   |   ANIMACY = I: + (0.0)
|   |   |   ANIMACY = NA:
|   |   |   |   PARANUM <= 0 : + (33.0/11.4)
|   |   |   |   PARANUM > 0 : - (465.0/29.1)
|   |   |   ANIMACY = C:
|   |   |   |   BOTH_PRONOUNS = I: + (0.0)
|   |   |   |   BOTH_PRONOUNS = C:
|   |   |   |   BOTH_PRONOUNS = NA:
|   |   |   |   |
|   PRO_EQUIV = I:
|   |   ALIAS = C:
|   |   |   BOTH_PROPER_NOUNS = I: - (53.0/1.4)
|   |   |   BOTH_PROPER_NOUNS = C:
|   |   |   |   ANTE_MED <= -0.5294 :
|   |   |   |   ANTE_MED > -0.5294 :
|   |   |   |   |
|   |   |   BOTH_PROPER_NOUNS = NA:
|   |   |   |   ANTE_MED <= 0.8 : - (17.0/1.3)
|   |   |   |   ANTE_MED > 0.8 : + (21.0/2.5)
|   |   ALIAS = I:
|   |   |   WORDS_STR = C:
|   |   |   |   NUMBER = NA: - (0.0)
|   |   |   |   NUMBER = I:
|   |   |   |   NUMBER = C:
|   |   |   |   |
|   |   |   WORDS_STR = I:
|   |   |   |   ANTE_MED <= 0.7894 :
|   |   |   |   ANTE_MED > 0.7894 :
|   |   |   |   |
```

Figure 6.2: Top portion of the best-performing decision tree classifier for the NPA-PER development data

rule, respectively, and are used to compute the confidence associated with the classification of a test instance.

As an example, the first rule says that two NPs are coreferent if they have a content word in common. Out of the 840 training instance this rule covers, 100 of them are erroneous. Intuitively, this rule is too general as far as positing coreference relationships are concerned, and hence it should not be surprising to see that the number of exceptions is fairly large. Nevertheless, from a machine learning perspective, this rule achieves a fairly high accuracy rate (88%). In general, RIPPER may induce a rule with an error rate as high as 50%.

Overall, the induced rules in this ruleset are linguistically intuitive. In particular, no "Incompatible" feature value appears in any of the positive rules.

Coreference rules can also be read off a decision tree in a fairly straightforward manner. As mentioned before, each path from the root to a leaf corresponds to a coreference rule. For instance, the topmost rule in the NPAPER classifier posits two NPs as coreferent if both are proper nouns and are the same string, but do not have an ancestor-descendent relationship according to WordNet. The two numbers in parentheses following a classification value indicate the number of training instances covered by the node and the number of erroneously covered instances, respectively.[2] Our clustering algorithms use these two numbers to compute the confidence associated with the class value assigned to a test instance.

Coreference features. The coreference classifiers show how the given features can be combined into propositional rules for determining whether two NPs are

[2]More precisely, the second number is the upper limit of the $n\%$ confidence interval of the number of errors made on the set of training instances covered by the corresponding leaf node. See Quinlan (1993) for details.

coreferent. From a linguistic point of view, it would also be informative to see how good a given feature is with respect to distinguishing coreferent and non-coreferent NPs. One way to measure the discriminating power of a feature is via its *information gain*.

Before defining information gain, we need to understand the notion of the *entropy* of a set of instances S. Given a set of possible class values C for an instance in S,

$$Entropy(S) := -\sum_{c \in C} p(c) \log_2 p(c).$$

It is easy to show that

$$0 \leq Entropy(S) \leq \log_2 |C|.$$

In particular,

$$Entropy(S) = \begin{cases} 0 & \text{if all instances in } S \text{ have the same class value} \\ \log_2 |C| & \text{if } \frac{|S|}{|C|} \text{ of the instances in } S \text{ have class } c \text{ for each } c \in C \end{cases}.$$

Hence, entropy can be viewed as a measure of the (im)purity of a set of instances S with respect to the class values of the instances. Now, we can define the *gain* of a set S and a feature f as follows:

$$Gain(S, f) = Entropy(S) - \sum_{v \in Values(f)} \frac{|S_v|}{|S|} Entropy(S_v),$$

where $Values(f)$ is the set of possible values of f and S_v is the subset of S for which f has value v. In other words, $Gain(S, f)$ is the difference between the entropy of S and the expected value of the entropy after S is partitioned by f. So, if $Gain(S, f)$ is large, then knowledge of the value of f is useful for reducing the entropy of S (i.e., f is informative with respect to distinguishing instances

of different classes). In fact, the *Gain* measure is used by the ID3 decision tree learner (Quinlan, 1986) for selecting which feature to branch on at each step of its iterative induction process.

Hence, we can get a better idea of the relative discriminating power of the coreference features by computing their gain values. Rows 1-10 of Tables 6.1 and 6.2 show the ten features with the largest gain value for four of our data sets, based on the training sets on which the best-performing coreference systems are obtained. (Note that we only consider nominal features in our computation for the sake of simplicity.) For reference, we have also included the entropy of the original data in row 11.

According to Table 6.1, the four most informative features (as measured by information gain) for both MUC data sets — WORD_OVERLAP, SOON_STR, SOON_STR_NONPRO, and WORDS_STR — are all lexical features. The next two on the list are WORDS_SUBSTR (lexical) and WNCLASS (semantic). Further down the list we see two more lexical features: PN_SUBSTR and PN_STR. So seven of the ten features on both lists are lexical in nature. One plausible reason for the large number of high-ranking lexical features is that these features can be computed with better accuracies than the other types of features (e.g., syntactic and semantic features). Hence, it is too premature to jump to the conclusion that lexical features are necessarily more informative than other types of features in the coreference feature set.

Table 6.2 shows the most informative features for the BNEWS and NPAPER feature sets. Note from row 11 that the original entropies of these two data sets are much lower than those of the MUC data sets owing to the highly skewed class distributions in the former: the ACE training instances are created with

Table 6.1: The ten nominal features with the largest information gain in the coreference feature set as measured on the MUC-6 and MUC-7 training data

	MUC-6		MUC-7	
	Feature	Gain	Feature	Gain
1	WORD_OVERLAP	0.5579	WORD_OVERLAP	0.5008
2	SOON_STR	0.5459	SOON_STR	0.4962
3	SOON_STR_NONPRO	0.5370	SOON_STR_NONPRO	0.4909
4	WORDS_STR	0.5309	WORDS_STR	0.4878
5	WORDS_SUBSTR	0.5231	WNCLASS	0.4818
6	WNCLASS	0.5218	WORDS_SUBSTR	0.4683
7	ALIAS	0.5188	AGREEMENT	0.4635
8	PN_SUBSTR	0.5153	PN_STR	0.4629
9	PN_STR	0.5095	PN_SUBSTR	0.4565
10	CONSTRAINTS	0.4937	PRO_RESOLVE	0.4561
11	(Original entropy)	(0.7926)	(Original entropy)	(0.6994)

Table 6.2: The ten nominal features with the largest information gain in the coreference feature set as measured on the BNEWS and NPAPER training data

	BNEWS		NPAPER	
	Feature	Gain	Feature	Gain
1	WNCLASS	0.0321	WNCLASS	0.0330
2	AGREEMENT	0.0259	AGREEMENT	0.0258
3	CONSTRAINTS	0.0189	SOON_STR	0.0243
4	SOON_STR	0.0182	WORD_OVERLAP	0.0203
5	BOTH_PRONOUNS	0.0165	SOON_STR_NONPRO	0.0202
6	PRO_EQUIV	0.0165	WORDS_STR	0.0195
7	GENDER	0.0136	PN_SUBSTR	0.0189
8	WORD_OVERLAP	0.0115	WORDS_SUBSTR	0.0187
9	PN_SUBSTR	0.0101	PN_STR	0.0180
10	WORDS_SUBSTR	0.0098	ALIAS	0.0158
11	(Original entropy)	(0.1521)	(Original entropy)	(0.1431)

McCarthy and Lehnert's method, whereas the MUC ones are created via Soon-1 and Soon-2. Unlike the MUC results in Table 6.1, the top two features on the list include a semantic feature (WNCLASS) and a syntactic feature (AGREEMENT). Nevertheless, a fair number of lexical features such as SOON_STR, WORD_OVERLAP, and PN_SUBSTR still appear in both lists.

Test accuracy of the coreference classifiers. Although we are primarily interested in the F-measure score achieved by a coreference system, it would be informative to compute the (instance-level) accuracy of the coreference classifier

Table 6.3: Test accuracy of the coreference classifier underlying the best-performing MUC-6 system and the effect of clustering

	Clustering	Accuracy on positive instances	Accuracy on negative instances	Overall accuracy
1	before	0.3415 (3583/10493)	0.9959 (596771/599225)	0.9846
2	after	0.8084 (8483/10493)	0.9625 (576732/599225)	0.9598

Table 6.4: Test accuracy of the coreference classifier underlying the best-performing MUC-7 system and the effect of clustering

	Clustering	Accuracy on positive instances	Accuracy on negative instances	Overall accuracy
1	before	0.4527 (1616/3570)	0.9922 (337772/340424)	0.9866
2	after	0.5569 (1988/3570)	0.9860 (335664/340424)	0.9816

from a machine learning perspective. Row 1 of Tables 6.3-6.7 reports the overall accuracy of the classifier underlying the best-performing coreference system on the test data, as well as its accuracies measured on just the positive and negative instances. As we can see, the accuracy rates on the positive instances range from 0.30 to 0.55, whereas those on the negative instances are always larger than 0.99. It should not be surprising to see the substantially higher accuracy rates on the negative instances: the fact that the negatives significantly outnumber the positives in the training sets causes the resulting classifiers to be biased in assigning a test instance the majority class label. For the same reason, the overall accuracy rates are very high (> 0.98).

Table 6.5: Test accuracy of the coreference classifier underlying the best-performing BNEWS system and the effect of clustering

	Clustering	Accuracy on positive instances	Accuracy on negative instances	Overall accuracy
1	before	0.3023 (2232/7383)	0.9958 (454298/456230)	0.9847
2	after	0.5578 (4118/7383)	0.9726 (443742/456230)	0.9660

Table 6.6: Test accuracy of the coreference classifier underlying the best-performing NPAPER system and the effect of clustering

	Clustering	Accuracy on positive instances	Accuracy on negative instances	Overall accuracy
1	before	0.3989 (5897/14785)	0.9934 (888719/894581)	0.9838
2	after	0.8239 (12181/14785)	0.9099 (813952/894581)	0.9085

Table 6.7: Test accuracy of the coreference classifier underlying the best-performing NWIRE system and the effect of clustering

	Clustering	Accuracy on positive instances	Accuracy on negative instances	Overall accuracy
1	before	0.3008 (2370/7879)	0.9982 (912529/914164)	0.9923
2	after	0.6203 (4887/7879)	0.9886 (903723/914164)	0.9854

Effect of clustering. Note that the accuracy rates reported in row 1 of Tables 6.3-6.7 are computed based on classification decisions that do not necessarily satisfy the transitivity constraint inherent in the coreference relation. To study the extent to which the classification decisions are altered by the clustering algorithm in the course of producing a partition, we measure the instance-level accuracy on the same test data *after* the aggressive-merge clustering algorithm is applied to coordinate these decisions. Results are shown in row 2 of Tables 6.3-6.7. Not surprisingly, the aggressive merging nature of the clustering algorithm causes the accuracy on the positive instances to rise dramatically (by 0.1-0.46). However, such increase is accomplished at the expense of the accuracy on the negative instances. This ultimately causes overall accuracy to deteriorate, as seen in the rightmost column of these tables.

6.1.2 Analyzing Anaphoricity Rules and Features

Anaphoricity rules. As with coreference resolution, we examine the anaphoricity classifiers to get an idea of what rules are important to anaphoricity determination. Statistics of the data sets on which the anaphoricity classifiers are trained are given in Table 6.8, and three of the resulting anaphoricity classifiers (one for each of MUC-7, BNEWS, and NPAPER) are shown in Figures 6.3-6.5. Note that these are the classifiers employed in the best-performing coreference systems and are therefore trained with different cost ratios (six for MUC-7, eight for BNEWS, and three for NPAPER). Because of the higher penalty placed on misclassified positive instances, it is now possible for RIPPER to induce rules with an error rate of larger than 50%. (See rule 6 in the MUC-7 classifier for an example.) Despite the difference in the cost ratio used to train these classifiers, we can see that

Table 6.8: Statistics of the data sets on which the anaphoricity classifiers are trained

Data set	Number of training insts	Number of positive insts	% positives
MUC-6	1035	1900	0.3526
MUC-7	1040	2421	0.3005
BNEWS	3242	9759	0.2494
NPAPER	4704	9772	0.3250
NWIRE	3201	12285	0.2067

```
+ 675 425 IF HEAD_MATCH = Y .
+ 35 6 IF PRONOUN = NOMINATIVE .
+ 3 0 IF DEFINITE = Y SUBCLASS = Y FIRST_PARA = Y .
+ 19 10 IF DEFINITE = Y SUBCLASS = Y THE_SING_N = Y .
+ 24 2 IF ALIAS = Y BARE_SINGULAR = Y .
+ 28 47 IF APPOSITIVE = Y .
+ 11 0 IF BARE_SINGULAR = N PRONOUN = POSSESSIVE .
+ 11 4 IF EMBEDDED = N PRONOUN = ACCUSATIVE .
- 1927 234 IF .
.
```

Figure 6.3: Anaphoricity classifier trained on the MUC-7 data ($cr = 6$)

the features STR_MATCH, HEAD_MATCH, PRONOUN, and BARE_SINGULAR are used extensively in all of them.

Anaphoricity features. To better understand which features are useful in distinguishing anaphoric and non-anaphoric NPs, we can compute the gain value for each of the anaphoricity features. Rows 1-10 of Tables 6.9 and 6.10 show the ten features with the largest gain value for four of our data sets, based on the training sets on which the best-performing coreference systems are obtained. As before, the entropy of the original data is given in row 11.

```
+ 1014 183 IF HEAD_MATCH = Y PROPER_NOUN = Y .
+ 429 70 IF HEAD_MATCH = Y PRONOUN = NOMINATIVE .
+ 306 56 IF STR_MATCH = Y EMBEDDED = Y .
+ 501 949 IF STR_MATCH = Y BARE_SINGULAR = N .
+ 129 95 IF PRONOUN = NOMINATIVE .
+ 603 3569 IF ARTICLE = DEFINITE BARE_SINGULAR = N .
- 4837 260 IF .
.
```

Figure 6.4: Anaphoricity classifier trained on the BNEWS data ($cr = 8$)

```
+ 1616 45 IF STR_MATCH = Y PROPER_NOUN = Y FIRST_SENT = N .
+ 638 89 IF STR_MATCH = Y EMBEDDED = Y .
+ 527 39 IF HEAD_MATCH = Y PRONOUN = NOMINATIVE .
+ 263 193 IF HEAD_MATCH = Y PROPER_NOUN = Y FIRST_SENT = N .
+ 300 378 IF HEAD_MATCH = Y THE_NE = Y .
+ 60 2 IF STR_MATCH = Y PRONOUN = ACCUSATIVE .
+ 124 39 IF PRONOUN = NOMINATIVE .
+ 155 155 IF APPOSITIVE = Y .
+ 72 134 IF STR_MATCH = Y PRONOUN = UNKNOWN .
+ 62 67 IF ARTICLE = DEFINITE BARE_SINGULAR = N UPPERCASE = Y .
+ 51 11 IF PRONOUN = ACCUSATIVE .
+ 17 9 IF THE_NE = Y SUBCLASS = Y THE_ADJ_N = N THE_N = N
   THE_SING_N = Y .
+ 21 40 IF SUBCLASS = Y DEFINITE = Y THE_N = Y .
+ 14 24 IF HEAD_MATCH = Y STR_MATCH = Y BARE_PLURAL = Y
   ARTICLE = DEFINITE .
- 8547 784 IF .
.
```

Figure 6.5: Anaphoricity classifier trained on the NPAPER data ($cr = 3$)

Table 6.9: The ten features with the largest information gain in the anaphoricity feature set as measured on the MUC-6 and MUC-7 training data

	MUC-6		MUC-7	
	Feature	Gain	Feature	Gain
1	HEAD_MATCH	0.2332	HEAD_MATCH	0.1522
2	STR_MATCH	0.2114	STR_MATCH	0.1327
3	PRONOUN	0.0576	PRONOUN	0.0826
4	ARTICLE	0.0522	BARE_SINGULAR	0.0267
5	UPPERCASE	0.0396	ARTICLE	0.0241
6	BARE_SINGULAR	0.0336	BARE_PLURAL	0.0168
7	PROPER_NOUN	0.0263	UPPERCASE	0.0160
8	HEADER	0.0251	THE_N	0.0156
9	BARE_PLURAL	0.0158	THE_SING_N	0.0142
10	QUANTIFIED	0.0151	NUMBER	0.0106
11	(Original entropy)	(0.9364)	(Original entropy)	(0.8819)

Table 6.10: The ten features with the largest information gain in the anaphoricity feature set as measured on the BNEWS and NPAPER training data

	BNEWS		NPAPER	
	Feature	Gain	Feature	Gain
1	STR_MATCH	0.1503	STR_MATCH	0.2276
2	HEAD_MATCH	0.1494	HEAD_MATCH	0.2183
3	PRONOUN	0.1293	PRONOUN	0.1129
4	ARTICLE	0.0534	PROPER_NOUN	0.0612
5	PROPER_NOUN	0.0422	ARTICLE	0.0652
6	BARE_SINGULAR	0.0421	UPPERCASE	0.0486
7	UPPERCASE	0.0317	BARE_PLURAL	0.0429
8	EMBEDDED	0.0262	BARE_SINGULAR	0.0390
9	BARE_PLURAL	0.0178	EMBEDDED	0.0302
10	QUANTIFIED	0.0156	NUMBER	0.0282
11	(Original entropy)	(0.8102)	(Original entropy)	(0.9097)

Not surprisingly, the three most informative features for these data sets — HEAD_MATCH, STR_MATCH, and PRONOUN — also appear frequently in the anaphoricity classifiers (see Figures 6.3-6.5). The high discriminating power of HEAD_MATCH and STR_MATCH is a probable consequence of the fact that an NP is likely to be anaphoric if there is a lexically similar noun phrase preceding it in the associated text. The informativeness of PRONOUN can also be expected: most pronominal NPs are anaphoric.

Other highly ranked features include ARTICLE, BARE_SINGULAR, BARE_PLURAL, PROPER_NOUN, and UPPERCASE. In particular, the informativeness of the first four of these indicates that the (in)definiteness of an NP is an important factor in determining its anaphoricity, which is consistent with our linguistic intuition.

Effect of the cost ratio on coreference performance. It would be interesting to examine the effect of *cr* on the performance of the coreference system. Figures 6.6 and 6.7 show the recall, precision, and F-measure curves produced by increasing *cr* from one to ten for the BNEWS and NPAPER development data sets.

As *cr* increases, recall rises and precision drops. This should not be surprising, since (1) increasing *cr* causes fewer anaphoric NPs to be misclassified and allows the coreference system to find a correct antecedent for some of them, and (2) decreasing *cr* causes more truly non-anaphoric NPs to be correctly classified and prevents the coreference system from resolving them.

Furthermore, the figures illustrate why a locally-optimized approach is less appealing than a globally-optimized one: in both graphs, F-measure reaches the minimum at $cr = 1$ (which corresponds to using the local approach). As can be seen, the poor F-measure is a result of the dramatic loss in recall. This implies

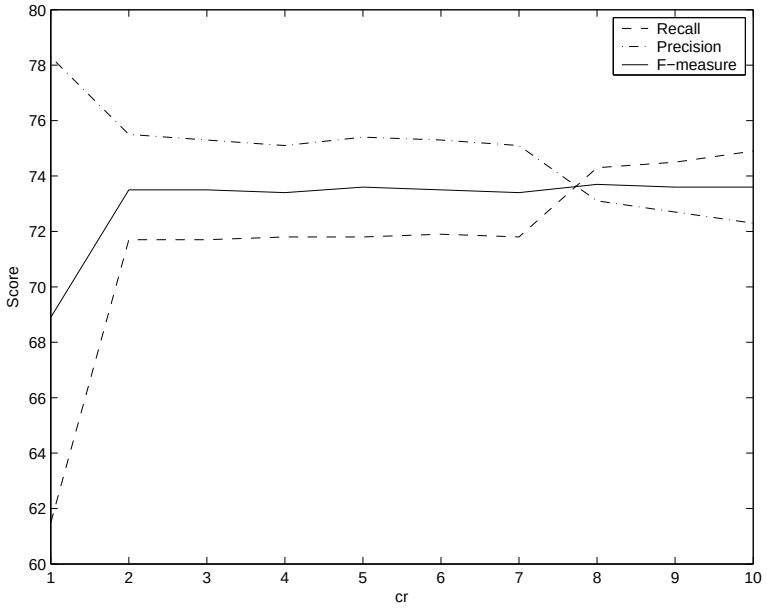

Figure 6.6: Effect of *cr* on the performance of the coreference system for the BNEWS development data

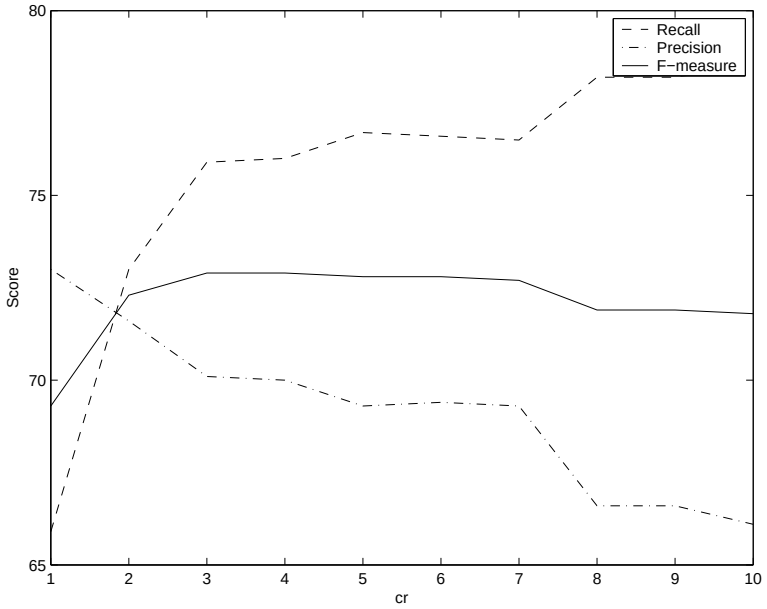

Figure 6.7: Effect of *cr* on the performance of the coreference system for the NPAPER development data

that many NPs are misclassified as non-anaphoric at this *cr* level.

Effect of *cr* on the accuracy of the anaphoricity classifiers. In Figures 6.6 and 6.7, we see that recall rises and precision drops as *cr* increases. We attribute the rise in recall to the increase in the accuracy of the underlying anaphoricity classifier on the positive instances; similarly, we attribute the drop in precision to the degradation in the accuracy of the classifier on the negative instances. To empirically validate our hypothesis, we show how the accuracies on the positive and negative anaphoricity instances as well as overall accuracy respond to changes in *cr* for the BNEWS and NPAPER development data sets in Figures 6.8 and 6.9. Comparing Figures 6.6 and 6.8, we see that the results are consistent with our hypothesis. Specifically, (1) the accuracy of the anaphoricity classifier on the positive instances is positively correlated with the recall of the coreference system, and (2) the accuracy of the classifier on the negative instances is also positively correlated with the precision of the coreference system. We can draw the same conclusions on the NPAPER development data by comparing Figures 6.7 and 6.9.

Test accuracy of the anaphoricity classifiers. Although we are primarily interested in the effect of anaphoricity determination on coreference resolution, it would also be informative to examine the accuracy of the anaphoricity classifiers on the test data. Table 6.11 shows the test accuracy of the anaphoricity classifier underlying the best-performing coreference system for each of our five data sets. As we can see, the accuracy rates on the positive (negative) instances are higher (lower) for the MUC-6, MUC-7, and BNEWS data sets in comparison to the NPAPER and NWIRE data sets. This can be attributed in part to the fact that the cost ratio used to train these classifiers are higher (6-8 as opposed to 3-4).

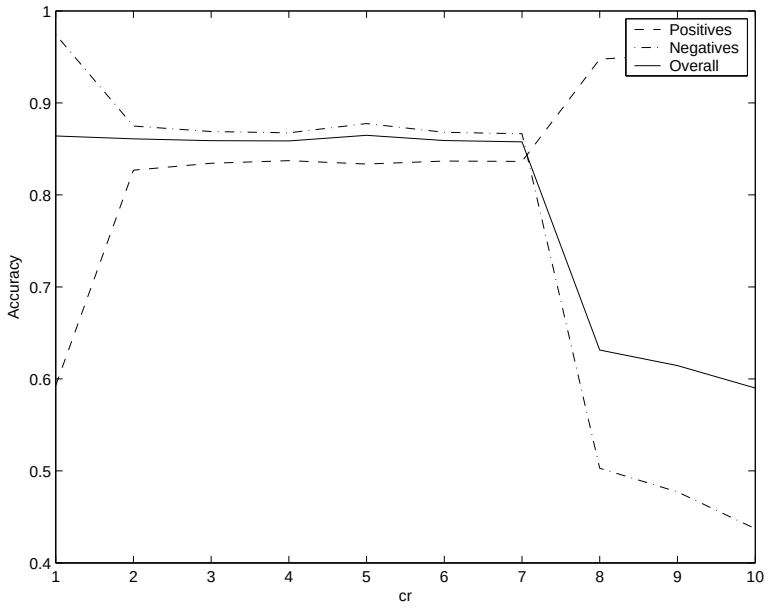

Figure 6.8: Effect of *cr* on the accuracy of the anaphoricity classifier for the BNEWS development data

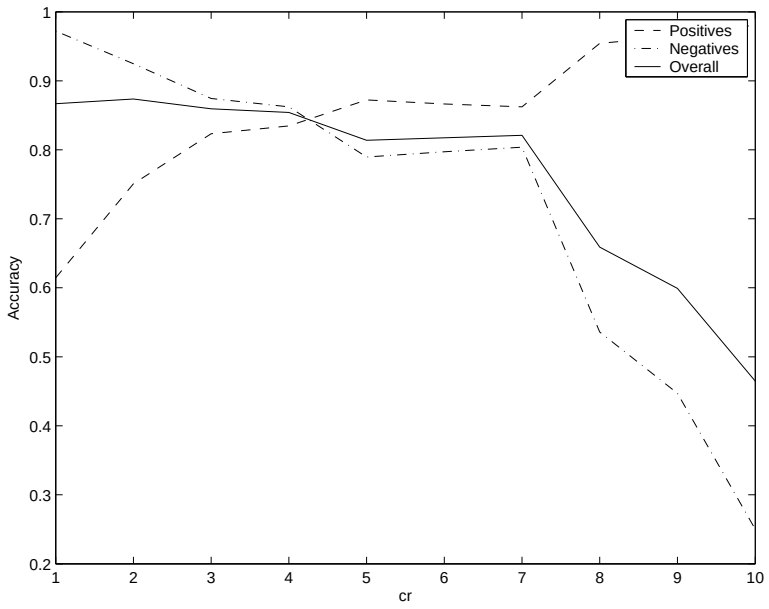

Figure 6.9: Effect of *cr* on the accuracy of the anaphoricity classifier for the NPA-PER development data

Table 6.11: Test accuracy of the anaphoricity classifiers underlying the best-performing coreference systems

Data set	*cr*	Accuracy on positive instances	Accuracy on negative instances	Overall accuracy
MUC-6	6	0.9051 (1373/1517)	0.6603 (1971/2985)	0.7428
MUC-7	6	0.8353 (776/929)	0.7359 (1864/2533)	0.7626
BNEWS	8	0.9032 (1307/1447)	0.5034 (2135/4241)	0.6051
NPAPER	3	0.8095 (1147/1417)	0.8516 (3046/3577)	0.8396
NWIRE	4	0.7441 (1012/1360)	0.8789 (4225/1360)	0.8492

Overall, there does not seem to be a universally good value of *cr*.

6.2 Sensitivity to Parameter Changes

Recall from Section 5.4 that we attribute the superior performance of our approach over existing approaches to the underlying parameterization, which presumably yields a search space containing high-performing systems. In this section, we want to get an idea of which values in the parameterization are essential to building a high-performing coreference system. Our investigation will proceed as follows.

6.2.1 Experimental Setup

Recall that a coreference system is represented as a 6-tuple in the space Λ comprising a *clustering algorithm* (C), a *training instance creation method* (I), a decision of whether to apply *rule pruning* (R), a *learning algorithm* (L), a *coreference feature set* (F), and a cost ratio specifying how to train an *anaphoricity classi-*

fier (A). Assume that we are given some training data D, and that the best-performing coreference system determined by **TuneCorefParams** based on D is $\lambda^* = < c,i,r,l,f,a >$, where $c \in C$, $i \in I$, $r \in R$, $l \in L$, $f \in F$, and $a \in A$. Now, we perform the following steps for each element e of λ^*.

1. Remove any coreference system $\lambda \in \Lambda$ such that e is an element of λ.

2. Run **TuneCorefParams** to re-tune the parameters of the coreference system given the reduced search space.

3. Evaluate the resulting coreference system on the test data and compare the result with that obtained using λ^*.

Our assumption underlying this set of experiments is that if excluding e from the search space causes system performance to deteriorate, then e is probably crucial to constructing a high-performing coreference system for the data set under consideration.[3]

6.2.2 Results and Discussion

In this subsection, we investigate the performance impact of removing each e from λ^* for each of our five coreference data sets.

Clustering algorithm. As seen in Section 5.4, all of the best-performing coreference systems make use of the aggressive-merge clustering algorithm. Hence, we will simply remove aggressive-merge from the set of clustering algorithms consid-

[3]Naturally, we expect the performance of the resulting system to deteriorate in comparison to λ^* because **TuneCorefParams** now considers a smaller number of coreference systems in the search space.

Table 6.12: Effect of perturbing each parameter value in the best-performing system for the MUC-6 test data. Row 1 shows the best-performing system; rows 2-7 show the re-trained systems with the perturbed parameter italized.

	F-measure	C	I	R	L	F	A
1	67.9	aggressive	Soon-2	yes	RIPPER	expanded	*cr*=6
2	65.6	*single-link*	Soon-1	no	C4.5	expanded	*cr*=6
3	66.4	aggressive	*Soon-4*	no	RIPPER	expanded	*cr*=4
4	67.9	aggressive	Soon-2	*no*	RIPPER	Soon	*cr*=6
5	67.8	aggressive	Soon-2	no	*C4.5*	expanded	*cr*=2
6	67.9	aggressive	Soon-2	no	RIPPER	*Soon*	*cr*=6
7	67.4	aggressive	Soon-2	yes	RIPPER	expanded	*none*

ered by **TuneCorefParams** and then re-tune the parameters of the coreference model on the same held-out data as before.

The re-tuned parameters and the F-measure scores achieved by the resulting coreference systems for the five test sets are shown in row 2 of Tables 6.12-6.16. For convenience, we have also included the best system configurations and the corresponding F-measure scores that we saw in Section 5.4 in row 1 of these tables. In comparison to the corresponding results in row 1, we see that F-measure drops precipitously by 2.5 (MUC-6), 1.1 (MUC-7), 5.1 (BNEWS), 4.4 (NPAPER), and 4.5 (NWIRE). These results suggest that aggressive-merge has contributed considerably to the overall performance of our best-performing systems.

Examining the re-tuned parameters, we see that there is no clear preference between using best-first and closest-first as the underlying clustering algorithm: the MUC systems favor the former, whereas the ACE systems always choose the

Table 6.13: Effect of perturbing each parameter value in the best-performing system for the MUC-7 test data. Row 1 shows the best-performing system; rows 2-7 show the re-trained systems with the perturbed parameter italized.

	F-measure	C	I	R	L	F	A
1	62.1	aggressive	Soon-1	no	C4.5	expanded	*cr*=6
2	61.0	*single-link*	Soon-1	no	C4.5	expanded	*cr*=6
3	63.4	aggressive	*Soon-2*	no	C4.5	expanded	*cr*=6
4	–	–	–	–	–	–	–
5	62.3	aggressive	Soon-1	no	*RIPPER*	expanded	*cr*=6
6	62.3	aggressive	Soon-4	no	RIPPER	*Soon*	*cr*=6
7	61.1	aggressive	Soon-1	no	C4.5	expanded	*none*

Table 6.14: Effect of perturbing each parameter value in the best-performing system for the BNEWS test data. Row 1 shows the best-performing system; rows 2-7 show the re-trained systems with the perturbed parameter italized.

	F-measure	C	I	R	L	F	A
1	65.4	aggressive	M&L	no	C4.5	expanded	*cr*=8
2	60.3	*best-first*	M&L	no	C4.5	expanded	*cr*=8
3	63.9	aggressive	*Soon-1*	no	C4.5	expanded	*cr*=5
4	–	–	–	–	–	–	–
5	59.9	aggressive	M&L	no	*RIPPER*	expanded	none
6	61.9	aggressive	Soon-5	no	C4.5	*Soon*	*cr*=5
7	65.3	aggressive	M&L	no	C4.5	expanded	*none*

Table 6.15: Effect of perturbing each parameter value in the best-performing system for the NPAPER test data. Row 1 shows the best-performing system; rows 2-7 show the re-trained systems with the perturbed parameter italized.

	F-measure	C	I	R	L	F	A
1	68.0	aggressive	M&L	no	C4.5	expanded	*cr*=3
2	63.6	*best-first*	Soon-3	no	C4.5	expanded	*cr*=3
3	65.7	aggressive	*Soon-3*	no	C4.5	expanded	*cr*=2
4	–	–	–	–	–	–	–
5	65.3	aggressive	Soon-1	no	*RIPPER*	expanded	*cr*=3
6	64.0	aggressive	Soon-1	no	C4.5	*Soon*	*cr*=2
7	67.2	aggressive	M&L	no	C4.5	expanded	*none*

Table 6.16: Effect of perturbing each parameter value in the best-performing system for the NWIRE test data. Row 1 shows the best-performing system; rows 2-7 show the re-trained systems with the perturbed parameter italized.

	F-measure	C	I	R	L	F	A
1	56.6	aggressive	M&L	no	C4.5	expanded	*cr*=4
2	52.1	*best-first*	M&L	no	C4.5	expanded	*cr*=2
3	52.8	aggressive	*Soon-1*	yes	RIPPER	expanded	*cr*=3
4	–	–	–	–	–	–	–
5	52.8	aggressive	Soon-1	yes	*RIPPER*	expanded	*cr*=3
6	51.2	aggressive	Soon-5	no	RIPPER	*Soon*	*cr*=2
7	53.5	aggressive	M&L	no	RIPPER	expanded	*none*

latter. In addition, while the re-tuning process yields different values of I, L, and A, the expanded feature set is always the preferred choice.

Training instance selection method. Now we consider removing the training instance creation methods that appear in the best parameter settings. Results are shown in row 3 of Tables 6.12-6.16. In comparison to the corresponding results in row 1, we see that F-measure drops by 1.5 (MUC-6), 1.5 (BNEWS), 2.3 (NPAPER), and 3.8 (NWIRE). Interestingly, F-measure rises by 1.3 for the MUC-7 data set; the unexpected performance gains can potentially be attributed to the presence of an insufficient amount of data for parameter tuning: the development data is too small to enable the determination of a good set of parameters.

For the remaining four data sets, we can see that the drop is less precipitous than when the clustering algorithm is altered (compare rows 2 and 3). This may be due to the fact that we have a family of instance creation methods that behave similarly to each other. In other words, even if we disallow the use of a particular training instance creation method, the re-tuning process may be able to replace it with a similar method from the remaining choices.

Rule pruning. As we can see from row 1 of Tables 6.12-6.16, only the best-performing system for the MUC-6 data set makes use of rule pruning. Hence, for the remaining four data sets, the best systems chosen by **TuneCorefParams** before and after removing the rule pruning option would be identical.

Now, focusing on the MUC-6 data set, we can see from rows 1 and 4 of Tables 6.12-6.16 that F-measure remains the same even if **TuneCorefParams** is not given the rule pruning option. Indeed, the "no rule pruning" system achieves this F-measure score with the Soon feature set. Therefore, we can conclude that neither

rule pruning nor the expanded feature set is needed to achieve an F-measure of 67.9 on the MUC-6 test data.

Learning algorithm. From row 1 of Tables 6.12-6.16, we can see that while the best-performing MUC-6 system uses RIPPER to train the underlying coreference classifier, C4.5 is the preferred learner for the remaining data sets. Given this, we force **TuneCorefParams** to use C4.5 when re-tuning the parameters for MUC-6. Similarly, RIPPER will be used for the remaining data sets during the re-tuning process.

Results are shown in row 5 of Tables 6.12-6.16. In comparison to the corresponding results in row 1, we see that the performance changes for the MUC data sets are fairly mild: F-measure drops by 0.1 for the MUC-6 data set and increases by 0.2 for the MUC-7 data set. On the other hand, we observe dramatic performance degradation in all ACE data sets: F-measure drops by 5.5, 2.7, and 3.8 for BNEWS, NPAPER, and NWIRE, respectively. These results seem to suggest that the choice of the learning algorithm is more of an issue for ACE than for MUC. Nevertheless, the consistently good performance achieved by employing C4.5 on all of these data sets makes it a better learner than RIPPER for machine learning for coreference resolution.

Feature set. Recall from row 1 of Tables 6.12-6.16 that the best-performing system for all five data sets rely on the expanded feature set. So now we simply force **TuneCorefParams** to use the Soon feature set during parameter re-tuning.

The performance of the resulting systems are shown in row 6 of Tables 6.12-6.16. In comparison to the corresponding results in row 1, we see that F-measure remains the same for the MUC-6 data set and even rises by 1.2 for the MUC-7 data

set. On the other hand, performance drops precipitously for the ACE data sets: F-measure decreases by 3.5, 4.0, and 5.4 for BNEWS, NPAPER, and NWIRE, respectively. These results provide empirical evidence that while the Soon feature set contains sufficient knowledge for the coreference system to achieve reasonably good performance on the MUC data sets, it fails to do so for the ACE data sets.

Anaphoricity determination. Results in row 1 of Tables 6.12-6.16 show that the best-performing coreference system for all five data sets rely on anaphoricity determination (although different values of *cr* are used). Unlike the other experiments in this subsection, in this case we simply deny the coreference system access to anaphoricity information. This would allow us to get an idea of the extent to which the anaphoricity determination component contributes to the overall system performance.

Results are shown in row 7 of Tables 6.12-6.16. In comparison to the corresponding results in row 1, F-measure drops by 0.5 (MUC-6), 1.0 (MUC-7), 0.1 (BNEWS), 0.8 (NPAPER), and 3.1 (NWIRE). Overall, adding anaphoricity determination yields a modest gain in performance.

6.2.3 Summary of Results

Summarizing the results in this section, we reach the following conclusions:

- The aggressive-merge clustering algorithm is always employed in the best-performing coreference systems.
- McCarthy and Lehnert's training instance creation method is essential to achieving good performance on the ACE data sets, whereas the Soon-1 and Soon-2 methods yield better-performing systems on the MUC data sets.

- Rule pruning does not appear to be a crucial parameter to **TuneCorefParams** as far as improving a coreference system is concerned.

- The robustness of C4.5 with respect to producing high-performing coreference systems makes it a better learning algorithm than RIPPER for learning-based coreference systems.

- The strong preference for the expanded feature set over the Soon feature set among the best-performing systems provides empirical evidence for the claim that incorporating additional knowledge (in the form of features) can improve the performance of a learning-based coreference system.

- Using automatically computed anaphoricity information to filter non-anaphoric NPs produces a modest gain in coreference performance.

6.3 Effect of Greedy Parameter Search

Because **TuneCorefParams** performs an exhaustive search in the parameter space, we are relieved of any search issues that we would have to deal with otherwise in order to make the most out of our approach to coreference resolution. However, as the parameter space becomes large (e.g., by increasing the size of an existing dimension or by increasing the dimensionality of the space), it may be too computationally expensive to perform an exhaustive search.

Hence, from a practical point of view, it would be informative to see if our approach could achieve a similar level of performance by employing a greedy search instead of an exhaustive search. In this section, we will study the trade-offs between performance and efficiency by comparing the results obtained via an exhaustive search and a greedy search in our coreference parameter space. Below we will first

describe our local search algorithm and then analyze the results obtained via this greedy approach.

6.3.1 The Greedy Search Algorithm

Our greedy search algorithm, **GreedyParamSearch**, is shown in Figure 6.10. **GreedyParamSearch** takes as input a training corpus T, a coreference scoring function S, a parametric family of coreference systems Λ, and a coreference system configuration $\lambda \in \Lambda$ that serves as a starting point in our greedy search. Like other greedy search algorithms, **GreedyParamSearch** attempts to make the best local move from the current position in each iteration, and terminates when the best move fails to improve performance.

There are two implementation issues we have to consider. First, we need to define what a local move is in our search space. This notion is made precise by the algorithm **Neighbors** (also shown in Figure 6.10), which computes the set of neighbors of a given coreference system configuration. Basically, $\alpha \in \Lambda$ is considered a neighbor of λ if and only if the corresponding vectors differ by at most one element (recall that each parameter setting is a 6-tuple).

Second, to determine the "best" move, we need a method for scoring a coreference system configuration. **GreedyParamSearch** does this in exactly the same way as **TuneCorefParams**. Specifically, the algorithm first divides the available documents in T into two sets, A and B. Given a coreference system configuration, the algorithm trains the underlying coreference and anaphoricity classifiers on A and then scores the resulting system on B according to the scoring function S.

GreedyParamSearch(T, S, Λ, λ_s)

Input: A training corpus T, a scoring function S, a parametric coreference system Λ, and a coreference system configuration $\lambda_s \in \Lambda$

Algorithm:
Randomly divide T into two disjoint sets A and B such that A is twice as large as B in terms of the number of documents.
$\lambda_{curr} := \lambda_s$
Train the underlying coreference and anaphoricity classifiers on A given λ_{curr}.
$Score(\lambda_{curr})$:= score of the resulting coreference system on B according to S.
do
  $BestScore := Score(\lambda_{curr})$
  $\lambda_{best} := \lambda_{curr}$
  foreach possible coreference parameter setting $\lambda_{new} \in$ **Neighbors**(λ_{curr},Λ) **do**
    Train the underlying coreference and anaphoricity classifiers on A given λ_{new}.
    $Score(\lambda_{new})$:= score of the resulting coreference system on B according to S.
  $\lambda^* := \arg\max_{\lambda \in \textbf{Neighbors}(\lambda_{new},\Lambda)} Score(\lambda)$
  $\lambda_{curr} := \lambda^*$
while $Score(\lambda^*) > BestScore$

Output: λ_{best}

Neighbors(α, Λ)

Input: A coreference system configuration α, and a parametric coreference system Λ

Algorithm:
$N := \emptyset$.
foreach possible coreference parameter setting $\lambda \in \Lambda$ **do**
  if α differs from λ along at most one dimension **do**
    $N := N \bigcup \{\lambda\}$
  endif

Output: N

Figure 6.10: The **GreedyParamSearch** Algorithm

6.3.2 Performance and Efficiency

We can now use **GreedyParamSearch** to greedily search for the best-performing coreference system on held-out data and apply the resulting system on our test data. In order to compare the performance of **GreedyParamSearch** with **TuneCorefParams**, we ensure that the conditions under which the two algorithms operate are identical. Specifically, **GreedyParamSearch** is given the same input parameters (i.e., T, S, and Λ) as **TuneCorefParams**, and the training corpus T is divided into a training sub-corpus (A) and a development sub-corpus (B) in the same way in both algorithms. As a starting point for the local search, we employ a coreference system configuration that has shown to perform reasonably well on the MUC data sets — the Soon et al. coreference system configuration.

Performance of greedy search. The best parameter setting found by **GreedyParamSearch** and the F-measure score achieved by the resulting system on the test data for each of the five data sets are shown in row 2 of Tables 6.17-6.21. For convenience, we have also included the best system configurations found by **TuneCorefParams** in row 1 of these tables. Comparing rows 1 and 2, we see that **GreedyParamSearch** finds the global maximum for the MUC-7 and BNEWS data sets. On the other hand, only a local maximum is reached for the remaining data sets: in comparison to the corresponding results in row 1, we see that F-measure drops slightly by 0.3 for the MUC-6 data set and dramatically by 4.0 and 5.4 for the NPAPER and NWIRE data sets, respectively.

It would be informative to start the greedy search from a different system configuration and see if we can get better (or worse) results. Comparing rows 1 and 2 of Tables 6.20 and 6.21 again, we conjecture that the poor results obtained

Table 6.17: Effect of greedy search on system performance for the MUC-6 test data. The F-measure score in row 1 is achieved by the best-performing system obtained via an exhaustive search. The scores in rows 2 and 3 are achieved by the best-performing systems obtained via a greedy search with different starting points.

	F-measure	C	I	R	L	F	A
1	67.9	aggressive	Soon-2	yes	RIPPER	expanded	cr=6
2	67.6	aggressive	Soon-2	no	C4.5	expanded	cr=6
3	67.6	aggressive	Soon-2	no	C4.5	expanded	cr=6

Table 6.18: Effect of greedy search on system performance for the MUC-7 test data. The F-measure score in row 1 is achieved by the best-performing system obtained via an exhaustive search. The scores in rows 2 and 3 are achieved by the best-performing systems obtained via a greedy search with different starting points.

	F-measure	C	I	R	L	F	A
1	62.1	aggressive	Soon-1	no	C4.5	expanded	cr=6
2	62.1	aggressive	Soon-1	no	C4.5	expanded	cr=6
3	62.1	aggressive	Soon-1	no	C4.5	expanded	cr=6

Table 6.19: Effect of greedy search on system performance for the BNEWS test data. The F-measure score in row 1 is achieved by the best-performing system obtained via an exhaustive search. The scores in rows 2 and 3 are achieved by the best-performing systems obtained via a greedy search with different starting points.

	F-measure	C	I	R	L	F	A
1	65.4	aggressive	M&L	no	C4.5	expanded	cr=8
2	65.4	aggressive	M&L	no	C4.5	expanded	cr=8
3	65.4	aggressive	M&L	no	C4.5	expanded	cr=8

Table 6.20: Effect of greedy search on system performance for the NPAPER test data. The F-measure score in row 1 is achieved by the best-performing system obtained via an exhaustive search. The scores in rows 2 and 3 are achieved by the best-performing systems obtained via a greedy search with different starting points.

	F-measure	C	I	R	L	F	A
1	68.0	aggressive	M&L	no	C4.5	expanded	cr=3
2	64.0	aggressive	Soon-1	no	C4.5	Soon	cr=2
3	63.2	single-link	Soon-1	no	C4.5	expanded	cr=3

Table 6.21: Effect of greedy search on system performance for the NWIRE test data. The F-measure score in row 1 is achieved by the best-performing system obtained via an exhaustive search. The scores in rows 2 and 3 are achieved by the best-performing systems obtained via a greedy search with different starting points.

	F-measure	C	I	R	L	F	A
1	56.6	aggressive	M&L	no	C4.5	expanded	cr=4
2	51.2	aggressive	Soon-5	no	RIPPER	Soon	cr=2
3	56.6	aggressive	M&L	no	C4.5	expanded	cr=4

via **GreedyParamSearch** may be attributable to the algorithm's inability to move to a configuration that employs the expanded feature set. Hence, in our second round of experiments, we give the algorithm an arguably better starting point — the Soon system configuration with the Soon feature set replaced with the expanded feature set. As before, the experiments are run under the same conditions in both the previous and current rounds.

Results are shown in row 3 of Tables 6.17-6.21. Here, we see that a global maximum is achieved for the NWIRE data set in addition to the MUC-7 and BNEWS data sets. In comparison to the corresponding results in row 2, we see that the algorithm reaches the same maximum as before for the MUC-6 data set. On the other hand, a different local maximum that yields even worse results is achieved for the NPAPER data set. Overall, these results are consistent with our intuition that we may end up at different critical points with different parameter initializations. Perhaps more importantly, **GreedyParamSearch** can yield coreference systems that perform substantially worse than those obtained via **TuneCorefParams**.

Table 6.22: The sequence of local moves made by the greedy search algorithm, with parameters initialized to Soon et al.'s system configuration

	MUC-6	MUC-7	BNEWS	NPAPER	NWIRE
1	expanded	expanded	M&L	aggressive	aggressive
2	cr=6	cr=6	aggressive	cr=2	cr=2
3	aggressive	aggressive	expanded	–	RIPPER
4	Soon-2	–	cr=8	–	Soon-5

Efficiency of greedy search. Recall that our primary goal in this section is to study the trade-offs between performance and efficiency by searching the parameter space (1) exhaustively and (2) greedily. Hence, an important question to ask at this point is: how many iterations does **GreedyParamSearch** take to reach the critical points?

We can answer this question by examining Tables 6.22 and 6.23, which show the steps **GreedyParamSearch** takes to reach a (local) maximum from our two starting points for the five data sets. Specifically, row i of each table shows which move was taken at the i-th iteration of the greedy search. For both of our starting points, **GreedyParamSearch** converges in at most four iterations for all of the data sets. Hence, despite its inferior performance to **TuneCorefParams**, the fast convergence of **GreedyParamSearch** makes it a viable alternative to **TuneCorefParams** when it is infeasible to search the parameter space exhaustively.

Table 6.23: The sequence of local moves made by the greedy search algorithm, with parameters initialized to Soon et al.'s system configuration except that the expanded feature set is used

	MUC-6	MUC-7	BNEWS	NPAPER	NWIRE
1	cr=6	cr=6	M&L	cr=3	M&L
2	aggressive	aggressive	aggressive	–	aggressive
3	Soon-2	–	cr=8	–	cr=4

6.4 Qualitative Error Analysis

Although our coreference systems performed well compared to existing systems, it is clear from the results that there is substantial room for improvement. In this section, we analyze the errors made by our coreference systems, with the goal of identifying new avenues for performance improvements.

Our qualitative error analysis is based on a collection of 25 test texts contributed equally by our five data sets (i.e., five texts chosen randomly from each data set). Since coreference performance is measured in terms of recall and precision, we divide the major sources of errors into three groups: (1) errors that affect both recall and precision; (2) errors that affect only precision; and (3) errors that affect only recall.

6.4.1 General Problems Affecting Recall and Precision

Errors in markable recognition. Remember that the maximum recall that can be achieved by a coreference system is limited by how well markables are extracted from the texts. Although our markable extraction algorithm has worked reasonably well (see Section 5.3), our error analysis indicates that a fairly large number of

markables are still missed. Worse still, many coreference relationships between the missing markables and the extracted ones can be recovered in a straightforward manner (e.g., by simple string matching facilities). Hence, we expect system recall to improve simply by improving the recall of the markable extraction algorithm.

On the other hand, erroneously extracted markables have caused system precision to drop. For instance, *head SEC* was (erroneously) extracted as a markable from the phrase *Washington to head SEC*. The markable was then (erroneously) merged with other NPs with the word *head* via the WORD_OVERLAP rule (see the first rule in Figure 6.1). Hence, we expect system precision to improve simply by improving the precision of the markable extraction algorithm.

Errors in feature value computation. Although errors in feature value computation is in general a problem for learning-based coreference systems, our analysis reveals that, for our five evaluation data sets, many recall and precision errors can be avoided by more accurate recognition of alias (in particular acronyms), appositives, and predicate nominal constructions. Hence, it is worth the effort of trying to improve accuracy in computing these features.

Similarly, errors in resolving pronouns and pronominal references have caused both system recall and precision to drop. In other words, while antecedents were found for non-anaphoric pronouns, many anaphoric pronouns remained unresolved. Moreover, the heuristics employed by our in-house pronoun resolution algorithm have proved insufficient for performing accurate pronoun resolution. The algorithm is especially weak in resolving pronouns in quoted speech, locative anaphors (e.g., *here*, *there*), demonstratives (e.g., *this*, *these*), and pronouns such as *we* that are used to refer to organizations (e.g., IBM) and locations (e.g., Europe). One

plausible way to improve the accuracy of pronoun resolution is to design a set of features specifically for acquiring pronoun resolution heuristics from annotated data instead of relying on the knowledge-based resolver.

6.4.2 Problems Affecting Precision

Induction of over-simplistic coreference rules. Over-simplistic coreference rules are rules that are too "lax" in positing two NPs as coreferent. As an example, consider again the best-performing coreference classifier for the MUC-6 data set (see Figure 6.1). The first rule in this classifier posits a coreference relationship between two NPs that have a content word in common. Note that this coreference constraint is too lax not only from an intuitive point of view but also from an empirical point of view: the rule is responsible for many erroneous mergings of NP clusters. Over-simplistic rules are likely to be induced when (1) high-recall, low-precision features are available to the learner (e.g., WORD_OVERLAP, SOON_STR) and (2) there is an insufficient number of negative training instances needed to induce high-precision rules (e.g., when Soon-1 or Soon-2 is used for creating training instances).

At first sight, it seems that this problem can simply be addressed either by removing high-recall, low-precision features from the feature set, or by using McCarthy and Lehnert's training creation method. Nevertheless, neither solution is particularly satisfactory for the following reasons. First, the removal of high-recall, low-precision features may adversely affect the recall and ultimately the overall performance of the system. Second, if the parameter tuning algorithm prefers Soon-1/Soon-2 over M&L, then the training data created by M&L is probably too skewed for the coreference classifier to perform well.

Instead, we propose the following solutions to the problem. Linguistically speaking, we can split a high-recall, low-precision feature into primitive features that are expected to be more precision-oriented than the original feature. For instance, we can split WORD_OVERLAP into two primitive features, one restricts the application of WORD_OVERLAP to head nouns and the other to prenominal modifiers. The reason is that word overlap between head nouns is intuitively a more reliable coreference indicator than word overlap between prenominal modifiers. Extra-linguistically speaking, we suggest devising a new training instance creation method in which we retain the set of positive instances created by Soon-1/Soon-2 but generate *additional* negative instances by adopting Soon-3's negative instance creation method, for example.

Aggressive NP merging. The aggressive-merge clustering algorithm encouraged many erroneous mergings of NPs. Many of these errors can be avoided by using precision-oriented clustering algorithms such as best-first and closest-first. However, the results in Tables 6.12-6.16 indicate that these two clustering algorithms have not performed as well as those employing aggressive-merge (in terms of F-measure). Because of the consistent preference for aggressive-merge by the best-performing systems, we suggest a precision-enhancing variant of aggressive-merge that adopts a more stringent merging criterion. Basically, for each NP_j, we still merge with each preceding coreferent NP, NP_i, as in aggressive-merge, but impose the additional check that each NP in the same cluster as NP_j is **compatible** with each NP in the same cluster as NP_i before merging. We can then define two NPs to be compatible if the probability that they are coreferent is greater than δ, for instance. This compatibility check is first applied to a closest-first clustering

algorithm in Cardie and Wagstaff's (1999) coreference system.

6.4.3 Problems Affecting Recall

Lack of an inference mechanism. Although the coreference feature set has a number of lexical features for performing string matching of varying degrees of sophistication, our coreference systems currently have a very limited capability for detecting co-referring NPs that are lexically dissimilar. Our analysis reveals that the recall of our systems is severely limited by their inability to detect these coreference relationships. Hence, to boost recall, the system must be equipped with the ability to perform the following kinds of inference.

- Detection of synonyms or near-synonyms (e.g., *the airline* and *the carrier*). Such detection becomes complicated for polysemous words such as *carrier*, in which case the sense of the word might have to be disambiguated first. Our SUBCLASS feature attempts to detect synonyms and near-synonyms with the help of WordNet in a context-independent manner (i.e., all senses of a word are taken into account during the detection). However, the feature does not appear to be useful to the learner, probably because of its context-independent nature.

- Detection of equivalent time expressions (e.g., *1995* and *last year*). Although our systems have employed a date normalizer to assist with the identification of equivalent time expressions, some of them remain undetected because of errors in the normalization process.

- Detection of subclass/part-whole coreference relationships (e.g., *spy satellites* and *remote-sensing instruments*). Clearly, this has to be done in a context-

dependent manner, since not all remote-sensing instruments are spy satellites, for instance. Although the SUBCLASS feature is responsible for detecting subclass relationships, it is not particularly useful to the learner, again probably because SUBCLASS operates in a context-independent manner.

- Resolution of definite descriptions to proper nouns (e.g., *SGI* and *the computer workstation maker*). Again, this has to be done in a context-dependent manner, since SGI is not the only computer workstation maker, for instance. Currently, our systems do not have a feature specifically designed to detect this kind of coreference relationship.

- Inference based on world knowledge (e.g., *China* and *Beijing*). Without access to world knowledge, a coreference system is unlikely to be able to detect this kind of coreference relationships. Currently, some world knowledge is provided by WordNet via the SUBCLASS feature (a WordNet path exists that establishes Beijing as the capital of China, for instance). Unfortunately, as mentioned above, this feature does not appear to be useful to the learner.

Lack of corpus-specific features. As mentioned before, different coreference corpora may be annotated using different guidelines. For instance, unlike MUC, ACE considers a country and the people living in the country to be coreferent (e.g., *Britain* and *British*). Although a learning-based coreference system can in principle acquire these corpus-specific constraints for annotated data, our analysis shows that a number of co-referring country/people pairs are missed by our systems. Hence, incorporating corpus-specific features may increase system recall.

6.5 Chapter Summary

To better understand the strengths and weaknesses of our approach to coreference resolution, we focused on four different issues in our analysis of its performance in this chapter. First, we examined the features and the learned rules that are important to coreference resolution and anaphoricity determination. Second, we empirically determined which parameter values in our best-performing coreference systems are crucial to their superior performance. Third, we examined how system performance would change if we employed a greedy search rather than an exhaustive search for the best system configuration in the parameter search space. Lastly, we conducted a qualitative analysis of the errors made by our systems and suggested how further performance improvements could be obtained.

CHAPTER 7

CONCLUSION

> *The area [anaphora and coreference resolution] is difficult but not intractable ... All we have to do is work more and hope for slow, but steady progress. We just have to be patient!* — Mitkov (2001)

In this dissertation, we have described the design, implementation, and evaluation of a new approach to coreference resolution that improves existing machine learning approaches to the problem. This chapter summarizes the contributions of our research, discusses our recent work on weakly supervised approaches to coreference resolution, and outlines possible future directions based on our work.

7.1 Summary of Contributions

The contribution of our research is five-fold.

A best-performing coreference resolution system. We presented a coreference system that outperforms the best existing learning-based coreference engine — the Soon et al. system — on five standard coreference data sets. Importantly, our superior performance is achieved *without* relying on additional labeled data, allowing us to conclude that our approach has made more effective use of the available training data than the Soon et al. system.

Linguistic extensions to existing machine learning approaches to coreference resolution. We proposed two types of linguistic modifications.

- **Large-scale expansion in the number and sophistication of coreference knowledge sources.** In contrast to existing learning-based coref-

erence systems, which rely on a fairly small set of surface-level features, we investigated a large-scale expansion in the number and sophistication of the features available to the learning algorithm. Our expanded feature set has played a crucial role in achieving good performance on the ACE data sets.

- **A new corpus-based approach to anaphoricity determination.** We presented a new supervised approach to identifying anaphoric and non-anaphoric noun phrases and showed how such anaphoricity information can be used to improve learning-based coreference systems. Incorporating anaphoricity information into the coreference system yields modest performance gains. Perhaps more importantly, such gains are achieved without the use of additional training data: both the anaphoricity classifiers and the coreference classifiers are trained on the same data.

Extra-linguistic modifications to existing learning approaches to coreference resolution. We proposed two types of extra-linguistic modifications.

- **Error-driven rule pruning.** To more tightly tie the classification- and clustering-level coreference decisions, we proposed an error-driven rule pruning algorithm that optimizes the coreference classifier with respect to the clustering-level coreference scoring function.

- **Augmenting existing system components with alternative implementations.** We generalized the standard machine learning approach by only requiring the specification of a *set* of learning algorithms, clustering algorithms, and training instance creation methods that are potentially useful for building a high-performing coreference engine at the time of system construction. In particular, augmenting existing system components with alternative

implementations has enabled us to delay the decisions of which learning algorithm, clustering algorithm, and training instance creation method to use until training time. This makes it possible to base such design decisions on the data set on which the coreference system is trained, thus allowing the system better capture the specificities of the data set.

A data-driven approach to system selection. Selecting the best coreference system configuration is non-trivial, since it has to take into account the interactions among the system components. We tackled this problem via a corpus-based approach, in which we employed an exhaustive search for the system that achieves the best performance on held-out data among all possible configurations.

The significance of this approach is three-fold. First, we showed that texts annotated with coreference information can be used not only for training classifiers but also for system selection. Second, we demonstrated how system performance can be optimized with respect to the given coreference scoring function, which is consistently ignored by existing approaches during system development. Lastly, we showed how to globally optimize a coreference system, unlike existing approaches in which the interactions among different components were seldom taken into account when design decisions were made.

Extensive performance analysis. We presented a detailed analysis of the performance of our approach, focusing on four different issues in our analysis. First, we analyzed the learned coreference and anaphoricity rules as well as the features that are important to these two tasks. Second, we empirically determined which parameter values in our best-performing coreference systems are crucial to their superior performance. Third, we examined how system performance would change

if we employed a greedy search rather than an exhaustive search for the best system configuration in the parameter search space. Finally, we conducted a qualitative analysis of the errors made by our systems and suggested how further performance improvements could be obtained.

7.2 Weakly Supervised Learning for Coreference Resolution

Recall that our approach currently uses $\frac{2}{3}$ of the available training documents for acquiring coreference classifiers and reserves the rest for tuning system parameters. Hence, given a fixed set of annotated documents, we will have more data for parameter tuning if we can reduce a learner's reliance on labeled data for classifier training.

Recently, a number of *weakly supervised* learning techniques have been developed, with the goal of reducing a learning algorithm's reliance on labeled data while maintaining a high level of accuracy. Given the observation that unlabeled data is inexpensive to obtain, these weakly supervised learning algorithms train a classifier on a small amount of labeled data and attempt to bootstrap the resulting classifier on a large amount of unlabeled data. In this section, we describe existing work as well as our related work on weakly supervised approaches to coreference resolution.

7.2.1 Previous Work

Müller et al. (2002) have applied a popular bootstrapping algorithm — *co-training* (Blum and Mitchell, 1998) — to the task of resolving anaphoric references in

German. This subsection first gives an overview of the co-training algorithm and then summarizes Müller et al.'s results.

The co-training algorithm trains two classifiers that can help augment each other's labeled data using two separate, but redundant *views* (or disjoint feature subsets) of the data. Initially, each classifier is trained using one view of the data and predicts the labels for all instances in the *data pool*, which consists of a randomly chosen subset of the unlabeled data. Each then independently selects its most confident predictions from the pool and adds the corresponding instances with their predicted labels to the labeled data. The number of instances to be added to the labeled data by each classifier at each iteration is limited by a pre-specified *growth size* to ensure that only the instances that have a high probability of being assigned the correct label are incorporated. The data pool is refilled with instances drawn from the unlabeled data and the process is repeated for several iterations.

Though the algorithm is conceptually simple, its theoretical guarantees come with two fairly strong assumptions on the views. First, each view must be sufficient for learning the concept. Second, the views must be conditionally independent of each other given the class label.[1]

Finding a pair of views that satisfies both of these conditions is a non-trivial problem for coreference resolution: neither can views be drawn from the left-hand and right-hand context as in part-of-speech tagging, nor can they be derived from features inside and outside the phrase under consideration as in named entity classification (Collins and Singer, 1999).

[1]Abney (2002) argues that the conditional independence assumption is remarkably strong and is rarely satisfied in real data sets, showing that a weaker independence assumption suffices.

Consequently, when applying co-training to resolving anaphoric references in German, Müller et al. propose a greedy algorithm for splitting the available features heuristically. They find that co-training shows no performance improvements for any type of German anaphor except pronouns over a baseline classifier trained on a small set of labeled data. They then conclude that co-training might only be useful for improving resolution performance on certain types of anaphora. However, they do not provide any analysis of why the task does not benefit from unlabeled data. In particular, the extent to which the two underlying assumptions on the views are satisfied is not known.

7.2.2 Our Related Work

As mentioned above, we have recently examined several issues in applying weakly supervised methods to coreference resolution (Ng and Cardie, 2003a,b). This subsection gives a summary of our related work and major findings.

Applying co-training with different view pairs. Müller et al.'s unsuccessful attempt in applying co-training to anaphora resolution may be attributed to the fact that the underlying view pair generated by their greedy method violates the sufficiency and/or conditional independence assumptions. As a result, we explore two additional heuristic methods for view factorization. The first method randomly divides the available features into two sets. The second splits the features according to the feature type, with the lexico-syntactic features forming one view and the remaining features forming the other.

Results on the MUC-6 and MUC-7 data sets indicate that co-training can improve coreference performance by as much as 5-10% in F-measure in comparison

to a baseline classifier trained on a small set of labeled data. However, consistent with the observations made by Nigam and Ghani (2000) and Pierce and Cardie (2001), we find that co-training is sensitive not only to the views employed, but to other input parameters such as the pool size, the growth size, and the number of iterations to run the algorithm as well. The lack of a principled method for determining these parameters in a weakly supervised setting where labeled data is scarce remains a serious disadvantage for co-training.

Investigating single-view algorithms as an alternative to co-training. Given the difficulty of view factorization for coreference resolution, we investigate *single-view* algorithms, which do *not* require view factorization, as an alternative to co-training for bootstrapping classifiers. In particular, we compare the performance of co-training with two commonly used single-view weakly supervised learners — self-training with bagging (Banko and Brill, 2001) and Expectation-Maximization (Nigam et al., 2000) — on the coreference task.

In self-training with bagging, we first employ bagging (Breiman, 1996) to train a committee of classifiers using the labeled data. Specifically, each classifier is trained on a *bootstrap sample* created by randomly sampling instances with replacement from the labeled data until the size of the bootstrap sample is equal to that of the labeled data. Then each member of the committee (or bag) predicts the labels of all unlabeled data. The algorithm selects an unlabeled instance for adding to the labeled data if and only if all bags agree upon its label. This ensures that only the unlabeled instances that have a high probability of being assigned the correct label will be incorporated into the labeled set. The above steps are repeated until all unlabeled data is labeled or a fixed point is reached.

Expectation-Maximization (EM) is first used as a single-view weakly supervised classification algorithm by Nigam et al. (2000). Like the classic unsupervised EM algorithm (Dempster et al., 1977), weakly supervised EM assumes a parametric model of data generation. The labels of the unlabeled data are treated as missing data. The goal is to find a model such that the posterior probability of its parameters is locally maximized given both the labeled data and the unlabeled data. Initially, the algorithm estimates the model parameters by training a probabilistic classifier on the labeled instances. Then, the *E-step* and the *M-step* are repeated for several iterations. In the E-step, all unlabeled data is probabilistically labeled by the classifier. In the M-step, the parameters of the generative model are re-estimated using both the initially labeled data and the probabilistically labeled data to obtain a maximum *a posteriori* hypothesis.

In comparison to co-training, self-training with bagging achieves substantially superior performance and is less sensitive to its input parameters. EM, on the other hand, fails to boost performance, and we attribute this phenomenon to the presence of redundant features in the underlying generative model. Consequently, we propose a wrapper-based feature selection method (Kohavi and John, 1997) for EM that results in performance improvements comparable to that observed with self-training. Overall, our results suggest that single-view weakly supervised learning algorithms are a viable alternative to co-training for coreference resolution where a natural feature split into separate, redundant views is not available.

Designing a single-view, multi-learner bootstrapping algorithm for coreference resolution. In addition to examining existing single-view weakly supervised learners such as self-training with bagging and EM, as described above, we

also develop a new single-view bootstrapping algorithm for coreference resolution. Motivated by the work of Goldman and Zhou (2000) and Steedman et al. (2003b), our bootstrapping algorithm uses *two different learning algorithms*, namely Naive Bayes (see Mitchell (1997), Chapter 6) and a decision list learner based on that described in Collins and Singer (1999), to train two classifiers on the *same* set of features (i.e., the full feature set). At each bootstrapping iteration, each classifier labels and scores all instances in the data pool. The highest scored instances labeled by one classifier are added to the training data of the other classifier and vice versa. Since the two classifiers are trained on the same view, it is important to maintain a separate training set for each classifier: this reduces the probability that the two classifiers converge to the same hypothesis at an early stage and hence implicitly increases the ability to bootstrap. The entire data pool is replenished with instances drawn from the unlabeled data after each iteration, and the process is repeated. Results on the MUC-6 and MUC-7 data sets indicate that our single-view, multi-learner bootstrapping algorithm can improve coreference performance by as much as 12-16% in F-measure in comparison to a baseline classifier trained on a small set of labeled data.

Developing a new method for ranking unlabeled instances. Although our single-view, multi-learner algorithm is fairly successful in bootstrapping coreference classifiers, we observe that the F-measure of the coreference system ultimately decreases as bootstrapping progresses. If the drop were caused by the degradation in the quality of the bootstrapped data, then a more "conservative" instance selection method than that employed by co-training would help alleviate this problem. Our hypothesis is that selection methods that are based solely on the confidence

assigned to an instance by a *single* classifier may be too liberal. In particular, these methods allow the addition of instances with opposing labels to the labeled data; this can potentially result in increased incompatibility between the classifiers.

Consequently, we develop a new procedure for ranking instances in the data pool to be fed back into the bootstrapping loop as labeled data. The bootstrapping algorithm then selects the highest ranked instances to add to the labeled data in each iteration. The method favors instances whose label is agreed upon by *both* classifiers. (**Preference 1**). However, incorporating instances that are confidently labeled by both classifiers may reduce the probability of acquiring new information from the data. Hence, the method imposes an additional preference for instances that are confidently labeled by one but not both. (**Preference 2**). If none of the instances receives the same label from the classifiers, the method resorts to the "rank-by-confidence" method employed by standard co-training (**Preference 3**).

More formally, define a binary classifier as a function that maps an instance to a value that indicates the probability that it is labeled as positive. Now, let μ be a function that rounds a number to its nearest integer. Given two binary classifiers C_1 and C_2 and instances i_1 and i_2, the ranking method shown in Figure 7.1 uses the three preferences described above to impose a partial ordering on the given instances for incorporation into C_2's labeled data. The method similarly ranks instances to be added to C_1's labeled data, with the roles of C_1 and C_2 reversed.

Cao et al. (2003) and Steedman et al. (2003a) have also independently investigated instance selection methods for co-training and verified the usefulness of ranking unlabeled instances using both classifiers involved in the bootstrapping loop. Interestingly (and incidentally), **Preference 2** forms the backbone of Cao et al.'s *uncertainty reduction* algorithm, where the highest ranking unlabeled

---

$i_1 > i_2$ if any of the following is true:

$[\mu(C_1(i_1)) = \mu(C_2(i_1))] \wedge [\mu(C_1(i_2)) \neq \mu(C_2(i_2))]$

$[\mu(C_1(i_1)) = \mu(C_2(i_1))] \wedge [\mu(C_1(i_2)) = \mu(C_2(i_2))] \wedge [|C_1(i_1) - C_2(i_1)| > |C_1(i_2) - C_2(i_2)|]$

$[\mu(C_1(i_1)) \neq \mu(C_2(i_1))] \wedge [\mu(C_1(i_2)) \neq \mu(C_2(i_2))] \wedge [\max(C_1(i_1), 1 - C_1(i_1)) > \max(C_1(i_2), 1 - C_1(i_2))]$

Figure 7.1: The ranking method that a binary classifier C_1 uses to impose a partial ordering on the instances to be selected and added to the training set of binary classifier C_2. i_1 and i_2 are arbitrary instances, and μ is a function that rounds a number to its closest integer.

instances are those that are most confidently labeled by one classifier and least confidently labeled by the other. **Preference 2** is also similar to Steedman et al.'s $S_{int\text{-}n}$ selection method, which selects an instance if it belongs to the intersection of the set of the n percent highest scoring instances of one classifier and the set of the n percent lowest scoring instances of the other. Nevertheless, to our knowledge, no previous work has examined a ranking method that combines the three preferences described above.

We compare the F-measure learning curves generated by our ranking method and the "rank-by-confidence" method employed in standard co-training for the MUC-6 and MUC-7 coreference data sets. Overall, the results are consistent with our intuition regarding the two methods. In the rank-by-confidence method, F-measure deteriorates in the course of bootstrapping as expected. On the other hand, our ranking method does not exhibit this performance trend.

7.3 Future Directions

There are numerous avenues for extending the current work, depending on whether one is primarily interested in making a linguistic or machine learning contribution to research in coreference resolution.

7.3.1 Potential Linguistic Extensions

Bootstrapping coreference knowledge. As discussed before, our coreference system lacks an inference mechanism for resolving lexically dissimilar NPs, in part owing to its lack of access to world knowledge as well as semantic knowledge for performing context-dependent word sense disambiguation and discourse analysis. Although it is important to identify new knowledge sources like these, the process

can sometimes be difficult, especially if the desired knowledge is too sophisticated to compute accurately. Perhaps a more appealing way to increase the knowledge of a coreference system is to have it discover new knowledge automatically by itself.

Harabagiu et al. (2001) present an algorithm for bootstrapping existing knowledge to improve the resolution of common nouns using the WordNet semantic knowledge base. The basic idea is to take advantage of the transitivity property inherent in the coreference relation. Given two common nouns C_1 and C_2, and two coreferent NPs A and B, knowing that C_1 is coreferent with A and and C_2 is coreferent with B allows the system to infer that C_1 is coreferent with C_2. Thus, new semantic consistency information can be discovered from the WordNet paths between C_1 and C_2.

Nevertheless, the problem of bootstrapping knowledge for coreference resolution is currently under-investigated. Given the initial success of Harabagiu et al.'s method in improving system recall, it seems that the problem warrants further investigation.

Adapting the system to handling spoken dialogues. So far we have only applied our system to resolving references in narrative texts. It would be interesting to investigate how well the system works on spoken dialogues. The presence of disfluencies and the abundance of references with non-NP antecedents in spoken dialogues not only distinguish them from written texts but also make the resolution process apparently more difficult.

There has been fairly little work on anaphora and coreference resolution for spoken dialogues. Most notably, Byron (2002) adopts a knowledge-based approach to resolving pronoun references in ten problem-solving dialogues from the TRAINS93

corpus (Heeman and Allen, 1995), achieving a reasonably good accuracy of 72%. However, this accuracy rate is achieved with two fairly strict assumptions. First, the resolution algorithm is assumed to have access to domain-specific semantic knowledge, which makes it difficult to port the system to new domains. Restricting the system's access to domain-independent resources, accuracy drops precipitously to 53%. Second, the reported accuracy is measured on anaphoric pronouns only. In other words, Bryon makes the practically unrealistic assumption that her system has access to perfect anaphoricity information.

More recently, Strube and Müller (2003) have adopted a machine learning approach to the problem. Their system achieves an F-measure of 47% in an evaluation on 20 Switchboard dialogues, subject to the strong assumption that the feature values are computed without any errors.

Overall, anaphora resolution for spoken dialogues is an under-investigated problem. It would therefore be interesting to see how to extend our system to handle pronoun references in spoken dialogues.

7.3.2 Potential Extra-Linguistic Extensions

Specializing coreference classifiers. The list of extensions to the standard machine learning framework that we proposed in this dissertation is by no means exhaustive. Other modifications are possible. For instance, we mentioned in Section 1.2 that coreference strategies often differ depending on the type of NP we are dealing with. This suggests that we may be able to improve system performance by training a separate coreference classifier for each type of NP: doing so may allow the learning algorithm to better capture the linguistic properties of each NP type.

Contrary to our expectation, however, preliminary experiments with the MUC

data sets indicate that training three separate classifiers (for pronouns, proper NPs, and common NPs) causes the F-measure of the coreference system to drop in comparison to training a single classifier. A closer examination of the results reveals the reason: some of the coreference rules induced in the single-classifier case are applicable to more than one NP type, but these rules are not always present in the corresponding NP-type-specific classifiers. This can be attributed in part to the smaller number of training instances used to acquire each NP-type-specific classifier in comparison to the single-classifier case: the reduction in the number of training instances causes certain linguistic patterns to occur too infrequently for the learning algorithm to account for.

One line of future work would be to examine whether the same conclusion can be drawn for the ACE data sets. It is possible that the coreference system may actually benefit from training multiple classifiers for ACE because of the availability of a larger amount of training data than MUC. Also, it would be worth investigating whether there are other ways of learning specialized coreference classifiers and combining their decisions.

Learning probabilistic relational models of coreference. One potential drawback of recasting coreference as a classification task, as we have seen, is that we can no longer enforce the transitivity constraint inherent in the coreference relation. This means that it is possible for a coreference system to determine that A is coreferent with B, and B with C, but that A and C are not coreferent.

Probabilistic relational models such as Relational Markov Networks (Taskar et al., 2002) have been developed to address this problem. These are undirected graphical models that allow classification decisions to be made in dependent re-

lation to each other by integrating information from individual entities as well as relations between them. Recently, McCallum and Wellner (2003) have developed a relational model specifically for proper noun coreference. The model is constructed as follows. First, we create a node for each entity in a text, connecting two nodes by an edge if we allow any form of dependency between them. Then, for each pair of nodes x_i, x_j in the graph, we create an additional node $y_{i,j}$ connected to x_i and x_j by an edge. These $y_{i,j}$ are binary-valued random variables used to indicate whether the NPs corresponding to x_i and x_j are coreferent. Next, we can define feature functions that are potentially useful for determining whether a pair of entities are coreferent or not over x_i, x_j and $y_{i,j}$. For instance, one feature function might have value 1 if and only if both NPs have the same gender and number and are coreferent with each other. These feature functions are combined into a probabilistic model having the following exponential form:

$$P(y \mid x) := \frac{1}{Z(x)} \exp\left(\sum_{i,j,l} \lambda_l \; f_l(x_i, x_j, y_{i,j})\right) \; ,$$

where $Z(x)$ is a normalization function, λ_l is a feature-weight parameter, and $f_l(x_i, x_j, y_{i,j})$ is a feature function associated with the parameter λ_l. After training the model parameters (i.e., the λ's) to maximize the likelihood of the given (labeled) data, we can apply the model to a new text to find the highest probability coreference solution, $y^* := \arg\max_y P(y \mid x)$, using standard inference procedures for relational models. It should be clear that the resulting solution does not make pairwise coreference decisions independently of each other. According to McCallum and Wellner (2003), results obtained via this model on proper noun coreference are promising. An interesting line of future work, then, would be to investigate if the model would work equally well on the full coreference task.

BIBLIOGRAPHY

Abney, Steven (2002). Bootstrapping. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics* (ACL-02), Philadelphia, PA, pp. 360–367. Association for Computational Linguistics.

Allen, James (1995). *Natural Language Understanding* (2nd ed.). The Benjamin/Cummings Publishing Company, Inc.

Anderberg, Michael (1973). *Cluster Analysis for Applications*. New York: Academic Press.

Aone, Chinatsu and Scott William Bennett (1995). Evaluating automated and manual acquisition of anaphora resolution strategies. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics* (ACL-95), Cambridge, MA, pp. 122–129. Association for Computational Linguistics.

Appelt, Douglas E. (1981). *Planning Natural-Language Utterances to Satisfy Multiple Goals*. Ph.D. thesis, Stanford University, Stanford, CA.

Azzam, Saliha (1996). Resolving anaphors in embedded sentences. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics* (ACL-96), Santa Cruz, CA, pp. 263–268. Association for Computational Linguistics.

Azzam, Saliha, Kevin Humphreys, and Robert Gaizauskas (1998). Evaluating a focus-based approach to anaphora resolution. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics* (COLING/ACL-98), Montreal, Canada, pp. 74–78. Association for Computational Linguistics.

Bagga, Amit and Breck Baldwin (1998). Entity-based cross-document coreferencing using the vector space model. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics* (COLING/ACL-98), Montreal, Canada, pp. 79–85. Association for Computational Linguistics.

Baldwin, Breck (1997). CogNIAC: High precision coreference with limited knowledge and linguistic resources. In *Proceedings of the ACL/EACL Workshop on Operational Factors in Practical, Robust Anaphora Resolution*, Madrid, Spain, pp. 38–45.

Baldwin, Breck and Thomas Morton (1998). Dynamic coreference-based summarization. In *Proceedings of the Third Conference on Empirical Methods in Natural Language Processing* (EMNLP-3), Granada, Spain. Association for Computational Linguistics.

Baldwin, Breck, Thomas Morton, Amit Bagga, Jason Baldridge, Raman Chandraseker, Alexis Dimitriadis, Kieran Snyder, and Magdalena Wolska (1998). Description of the UPenn CAMP system as used for coreference. In *Proceedings of the Seventh Message Understanding Conference* (MUC-7), San Francisco, CA. Morgan Kaufmann.

Banko, Michele and Eric Brill (2001). Scaling to very very large corpora for natural language disambiguation. In *Proceedings of the 39th Annual Meeting and 10th Conference of the European Chapter of the Association for Computational Linguistics* (ACL/EACL-01), Toulouse, France, pp. 26–33. Association for Computational Linguistics.

Bansal, Nikhil, Avrim Blum, and Shuchi Chawla (2002). Correlation clustering. In *Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science* (FOCS-02), Vancouver, Canada, pp. 238–247. IEEE Computer Society.

Bean, David and Ellen Riloff (1999). Corpus-based identification of non-anaphoric noun phrases. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics* (ACL-99), College Park, MD, pp. 373–380. Association for Computational Linguistics.

Berger, Adam L., Stephen A. Della Pietra, and Vincent J. Della Pietra (1996). A maximum entropy approach to natural language processing. *Computational Linguistics 22*(1), 39–71.

Bikel, Daniel M., Richard Schwartz, and Ralph M. Weischedel (1999). An algorithm that learns what's in a name. *Machine Learning: Special Issue on Natural Language Learning 34*(1–3), 211–231.

Blum, Avrim and Pat Langley (1997). Selection of relevant features and examples in machine learning. *Artificial Intelligence 97*(1–2), 245–271.

Blum, Avrim and Tom Mitchell (1998). Combining labeled and unlabeled data with co-training. In *Proceedings of the 11th Annual Conference on Computational Learning Theory* (COLT-98), Madison, WI, pp. 92–100.

Botley, Simon and Anthony McEnery (2000). Discourse anaphora: The need for synthesis. In S. Botley and T. McEnery (Eds.), *Corpus-based and Computational Approaches to Discourse Anaphora*, pp. 1–41. UCL Press.

Breck, Eric, John Burger, Lisa Ferro, David House, Marc Light, and Inderjeet

Mani (1999). A sys called Qanda. In *Proceedings of the Eighth Text REtrieval Conference TREC 8*, Gaithersburg, MD. NIST Special Publication.

Breiman, Leo (1996). Bagging predictors. *Machine Learning 24*, 123–140.

Brennan, Susan E., Marilyn Walker Friedman, and Carl J. Pollard (1987). A centering approach to pronouns. In *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics* (ACL-87), Stanford, CA, pp. 155–162. Association for Computational Linguistics.

Byron, Donna K. (2002). Resolving pronominal reference to abstract entities. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics* (ACL-02), Philadelphia, PA, pp. 80–87. Association for Computational Linguistics.

Cao, Yunbo, Hang Li, and Li Lian (2003). Uncertainty reduction in collaborative bootstrapping: Measure and algorithm. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics* (ACL-03), Sapporo, Japan, pp. 327–334. Association for Computational Linguistics.

Carbonell, Jaime and Ralf Brown (1988). Anaphora resolution: A multi-strategy approach. In *Proceedings of the 12th International Conference on Computational Linguistics* (COLING-88), Budapest, Hungary, pp. 96–101.

Cardie, Claire (1997). Empirical methods in information extraction. *AI Magazine 18*(4), 65–79.

Cardie, Claire and Nicholas Howe (1997). Improving minority class prediction using case-specific feature weights. In D. Fisher (Ed.), *Proceedings of the Four-*

teenth International Conference on Machine Learning (ICML-97), Vanderbilt University, Memphis, TN, pp. 57–65. Morgan Kaufmann.

Cardie, Claire and David Pierce (1998). Error-driven pruning of treebank grammars for base noun phrase identification. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics* (COLING/ACL-98), Montreal, Canada, pp. 218–224. Association for Computational Linguistics.

Cardie, Claire and Kiri Wagstaff (1999). Noun phrase coreference as clustering. In *Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora* (EMNLP/VLC-99), University of Maryland, College Park, MD, pp. 82–89. Association for Computational Linguistics.

Charniak, Eugene (1972). *Towards a Model of Children's Story Comphrension.* AI-TR 266, Artificial Intelligence Laboratory, Massachusetts Institute of Technology.

Charniak, Eugene (2000). A maximum-entropy-inspired parser. In *Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics* (NAACL-00), Seattle, WA, pp. 132–139. Association for Computational Linguistics.

Chomsky, Noam (1988). *Language and Problems of Knowledge. The Managua Lectures.* Cambridge, Massachusetts: MIT Press.

Cohen, William (1995). Fast effective rule induction. In *Proceedings of the Twelfth International Conference on Machine Learning* (ICML-95), Tahoe City, CA, pp. 155–123. Morgan Kaufmann.

Collins, Michael and Yoram Singer (1999). Unsupervised models for named entity classification. In *Proceedings of the 1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora* (EMNLP/VLC-99), University of Maryland, College Park, MD, pp. 100–110. Association for Computational Linguistics.

Connolly, Dennis, John D. Burger, and David S. Day (1994). A machine learning approach to anaphoric reference. In *Proceedings of International Conference on New Methods in Language Processing*, pp. 255–261.

Cristianini, Nello and John Shawe-Taylor (2000). *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge University Press.

Dale, Robert (1992). *Generating Referring Expressions: Constructing Descriptions in a Domain of Objects and Processes*. MIT Press.

Dempster, Arthur (1968). A generalization of Bayesian inference. *Journal of the Royal Statistical Society 30*, 205–247.

Dempster, Arthur P., Nan M. Laird, and Donald B. Rubin (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B 39*(1), 1–38.

Denber, Michel (1998). Automatic resolution of anaphora in English. Technical report, Eastman Kodak Co.

Evans, Richard (2001). Applying machine learning toward an automatic classification of *it*. *Literary and Linguistic Computing 16*(1), 45–57.

Fawcett, Tom (1996). *Learning with skewed class distributions — summary of responses*. Machine Learning List: Vol. 8, No. 20.

Fellbaum, Christiane (1998). *WordNet: An Electronic Lexical Database*. Cambridge, MA: MIT Press.

Gaizauskas, Robert, Takahiro Wakao, Kevin Humphreys, Hamish Cunningham, and Yorick Wilks (1995). Description of the LaSIE ystem as used for MUC-6. In *Proceedings of the Sixth Message Understanding Conference* (MUC-6), Columbia, MD, pp. 207–220. Morgan Kaufmann.

Ge, Niyu, John Hale, and Eugene Charniak (1998). A statistical approach to anaphora resolution. In E. Charniak (Ed.), *Proceedings of the Sixth Workshop on Very Large Corpora* (WVLC-98), Montreal, Canada, pp. 161–170. Association for Computational Linguistics.

Goldman, Sally and Yan Zhou (2000). Enhancing supervised learning with unlabeled data. In *Proceedings of the Seventeenth International Conference on Machine Learning* (ICML-00), Stanford, CA, pp. 327–334. Morgan Kaufmann.

Gordon, Diana F. and Donald Perlis (1989). Explicitly biased generalization. *Computational Intelligence 5*, 67–81.

Grishman, Ralph (1995). The NYU system for MUC-6 or Where's the syntax? In *Proceedings of the Sixth Message Understanding Conference* (MUC-6), Columbia, MD, pp. 167–175. Morgan Kaufmann.

Grosz, Barbara J. (1977). The representation and use of focus in a system for understanding dialogs. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence* (IJCAI-77), Cambridge, MA, pp. 67–76.

Grosz, Barbara J., Aravind K. Joshi, and Scott Weinstein (1995). Centering: A framework for modeling the local coherence of discourse. *Computational Linguistics 21*(2), 203–226.

Grosz, Barbara J. and Candace L. Sidner (1986). Attention, intentions, and the structure of discourse. *Computational Linguistics 12*(3), 175–204.

Haegeman, Liliane (1994). *Introduction to Government and Binding Theory* (2nd ed.). Blackwell Publishers Ltd.

Harabagiu, Sanda, Răzvan Bunescu, and Steven Maiorano (2001). Text and knowledge mining for coreference resolution. In *Proceedings of the 2nd Meeting of the North American Chapter of the Association for Computational Linguistics* (NAACL-01), Pittsburgh, PA, pp. 55–62. Association for Computational Linguistics.

Hawkins, John A. (1978). *Definiteness and Indefiniteness*. London: Croom Helm.

Heeman, Peter and James Allen (1995). The TRAINS spoken dialog corpus. CD-ROM, Linguistic Data Consortium.

Hirst, Graeme (1981). *Anaphora in Natural Language Understanding*. Springer Verlag.

Hobbs, Jerry (1978). Resolving pronoun references. *Lingua 44*, 311–338.

Iida, Ryu, Kentaro Inui, Hiroya Takamura, and Yuji Matsumoto (2003). Incorporating contextual cues in trainable models for coreference resolution. In *Proceedings of the EACL Workshop on The Computational Treatment of Anaphora*, Budapest, Hungary.

Kameyama, Megumi (1998). Intrasentential centering: A case study. In M. Walker, A. Joshi, and E. Prince (Eds.), *Centering Theory in Discourse*, pp. 89–112. Oxford University Press.

Kehler, Andrew (1997a). Current theories of centering for pronoun interpretation: A critical evaluation. *Computational Linguistics 23*(3), 467–475.

Kehler, Andrew (1997b). Probabilistic coreference in information extraction. In C. Cardie and R. Weischedel (Eds.), *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing* (EMNLP-2), Providence, RI, pp. 163–173. Association for Computational Linguistics.

Kennedy, Christopher and Branimir Boguraev (1996). Anaphor for everyone: Pronominal anaphora resolution without a parser. In *Proceedings of the 16th International Conference on Computational Linguistics* (COLING-96), Copenhagen, Denmark, pp. 113–118.

Kohavi, Ron and George H. John (1997). Wrapper for feature subset selection. *Artificial Intelligence Journal 96*(1–2), 273–324.

Kruskal, Joseph B. (1999). An overview of sequence comparison. In D. Sankoff and J. Kruskal (Eds.), *Time Warps, String Edits and Macromolecules: Theory and Practice of Sequence Comparison*. CSLI Publications and Cambridge University Press.

Kubat, Miroslav and Stan Matwin (1997). Addressing the curse of imbalanced training sets: One-sided selection. In *Proceedings of the 14th International Conference on Machine Learning* (ICML-97), Vanderbilt University, Memphis, TN, pp. 179–186. Morgan Kaufmann.

Lappin, Shalom and Herbert Leass (1994). An algorithm for pronominal anaphora resolution. *Computational Linguistics 20*(4), 535–562.

Lin, Dekang (1995). University of Manitoba: Description of the PIE system as used for MUC-6. In *Proceedings of the Sixth Message Understanding Conference* (MUC-6), Columbia, MD, pp. 113–126. Morgan Kaufmann.

Lin, Dekang (1998). Dependency-based evaluation of MINIPAR. In *Proceedings of the LREC Workshop on the Evaluation of Parsing Systems*, Granada, Spain, pp. 48–56.

Luo, Xiaoqiang, Abe Ittycheriah, Hongyan Jing, Nanda Kambhatla, and Salim Roukos (2004). A mention-synchronous coreference resolution algorithm based on the bell tree. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics* (ACL-04), Barcelona, Spain. Association for Computational Linguistics.

Mani, Inderjeet and George Wilson (2000). Robust temporal processing of news. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics* (ACL-00), Hong Kong, pp. 69–76. Association for Computational Linguistics.

Marcus, Mitchell P., Beatrice Santorini, and Mary Ann Marcinkiewicz (1993). Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics 19*(2), 313–330.

McCallum, Andrew and Ben Wellner (2003). Toward conditional models of identity uncertainty with application to proper noun coreference. In *Proceedings of the IJCAI Workshop on Information Integration on the Web*, Acapulco, Mexico.

McCarthy, Joseph and Wendy Lehnert (1995). Using decision trees for coreference resolution. In C. Mellish (Ed.), *Proceedings of the Fourteenth International Conference on Artificial Intelligence* (IJCAI-95), Montreal, Canada, pp. 1050–1055.

McCoy, Kathleen F. and Michael Strube (1999). Generating anaphoric expressions: Pronoun or definite description? In *Proceedings of the ACL Workshop on the Relation of Discourse/Dialogue Structure and Reference*, College Park, MD, pp. 63–71.

McKeown, Kathleen Rose (1985). *Text Generation: Using Discourse Strategies and Focus Constraints to Generate Natural Language Text*. Cambridge University Press.

Mitchell, Tom M. (1997). *Machine Learning*. New York: McGraw-Hill.

Mitkov, Ruslan (1997). Factors in anaphora resolution: they are not the only things that matter. A case study based on two different approaches. In *Proceedings of the ACL/EACL Workshop on Operational Factors in Practical, Robust Anaphora Resolution*, Madrid, Spain, pp. 14–21.

Mitkov, Ruslan (1998). Robust pronoun resolution with limited knowledge. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics* (COLING/ACL-98), Montreal, Canada, pp. 869–875. Association for Computational Linguistics.

Mitkov, Ruslan (1999). Anaphora resolution: The state of the art. Technical

Report (Based on the COLING/ACL-98 tutorial on anaphora resolution), University of Wolverhampton, Wolverhampton.

Mitkov, Ruslan (2001). Outstanding issues in anaphora resolution. In A. Gelbukh (Ed.), *Computational Linguistics and Intelligent Text Processing*, pp. 110–125. Springer.

Mitkov, Ruslan, Branimir Boguraev, and Shalom Lappin (2001). Introduction to the special issue on computational anaphora resolution. *Computational Linguistics 27*(4), 473–477.

Mitkov, Ruslan, Richard Evans, and Constantin Orasan (2002). A new, fully automatic version of Mitkov's knowledge-poor pronoun resolution method. In A. Gelbukh (Ed.), *Computational Linguistics and Intelligent Text Processing*, pp. 169–187. Springer.

Morton, Thomas (2000). Coreference for NLP applications. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics* (ACL-00), Hong Kong. Association for Computational Linguistics.

MUC-6 (1995). *Proceedings of the Sixth Message Understanding Conference* (MUC-6). San Francisco, CA: Morgan Kaufmann.

MUC-7 (1998). *Proceedings of the Seventh Message Understanding Conference* (MUC-7). San Francisco, CA: Morgan Kaufmann.

Müller, Christoph, Stefan Rapp, and Michael Strube (2002). Applying co-training to reference resolution. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics* (ACL-02), Philadelphia, PA, pp. 352–359. Association for Computational Linguistics.

Ng, Vincent (2004). Learning anaphoricity information to improve coreference resolution: Issues in representation and optimization. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics* (ACL-04), Barcelona, Spain. Association for Computational Linguistics.

Ng, Vincent and Claire Cardie (2002a). Combining sample selection and error-driven pruning for machine learning of coreference rules. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing* (EMNLP-02), Philadelphia, PA, pp. 55–62. Association for Computational Linguistics.

Ng, Vincent and Claire Cardie (2002b). Identifying anaphoric and non-anaphoric noun phrases to improve coreference resolution. In *Proceedings of the 19th International Conference on Computational Linguistics* (COLING-02), Taipei, Taiwan, pp. 730–736.

Ng, Vincent and Claire Cardie (2002c). Improving machine learning approaches to coreference resolution. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics* (ACL-02), Philadelphia, PA, pp. 104–111. Association for Computational Linguistics.

Ng, Vincent and Claire Cardie (2003a). Bootstrapping coreference classifiers with multiple machine learning algorithms. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, Sapporo, Japan, pp. 103–110. Association for Computational Linguistics.

Ng, Vincent and Claire Cardie (2003b). Weakly supervised natural language learning without redundant views. In *Human Language Technology Conference of*

the North American Chapter of the Association for Computational Linguistics: Main Proceedings (HLT/NAACL-03), Edmonton, Canada, pp. 173–180. Association for Computational Linguistics.

Nigam, Kamal and Rayid Ghani (2000). Analyzing the effectiveness and applicability of co-training. In *Proceedings of the Ninth International Conference on Information and Knowledge Management* (CIKM-2000), McLean, VA, pp. 86–93.

Nigam, Kamal, Andrew McCallum, Sabastian Thrun, and Tom Mitchell (2000). Text classification from labeled and unlabeled documents using EM. *Machine Learning 39*(2/3), 103–134.

Paice, Chris and Gareth Husk (1987). Towards the automatic recognition of anaphoric features in English text: the impersonal pronoun 'it'. *Computer Speech and Language 2*.

Pazzani, Michael, Christopher Merz, Patrick Murphy, Kamal Ali, Timothy Hume, and Clifford Brunk (1994). Reducing misclassification costs. In *Proceedings of the Eleventh International Conference on Machine Learning* (ICML-94), Rutgers University, New Brunswick, NJ, pp. 217–225. Morgan Kaufmann.

Pierce, David and Claire Cardie (2001). Limitations of co-training for natural language learning from large datasets. In L. Lee and D. Harman (Eds.), *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing* (EMNLP-01), Pittsburgh, PA, pp. 1–9. Association for Computational Linguistics.

Prince, Ellen (1981). Toward a taxonomy of given-new information. In P. Cole (Ed.), *Radical Pragmatics*, pp. 223–255. New York, N.Y.: Academic Press.

Quinlan, J. Ross (1986). Induction of decision trees. *Machine Learning 1*, 81–106.

Quinlan, J. Ross (1993). *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann.

Salton, Gerard, A. Wong, and C. S. Yang (1975). A vector space model for information retrieval. *Communications of the ACM 18*(11), 613–620.

Sidner, Candace (1979). *Towards a Computational Theory of Definite Anaphora Comprehension in English Discourse*. Ph.D. thesis, Massachusetts Institute of Technology.

Sidner, Candace (1981). Focusing for interpretation of pronouns. *American Journal of Computational Linguistics 7*, 217–231.

Singhal, Amit, Steve Abney, Michiel Bacchiani, Michael Collins, Donald Hindle, and Fernando Pereira (1999). AT&T at TREC-8. In *Proceedings of the Eighth Text REtrieval Conference TREC 8*, Gaithersburg, MD. NIST Special Publication.

Soderland, Stephen (1997). Learning to extract text-based information from the World Wide Web. In *Proceedings of Third International Conference on Knowledge Discovery and Data Mining* (KDD-97), Newport Beach, CA, pp. 251–254. AAAI Press.

Soon, Wee Meng, Hwee Tou Ng, and Daniel Chung Yong Lim (2001). A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics 27*(4), 521–544.

Steedman, Mark, Rebecca Hwa, Stephen Clark, Miles Osborne, Anoop Sarkar, Julia Hockenmaier, Paul Ruhlen, Steven Baker, and Jeremiah Crim (2003a). Example selection for bootstrapping statistical parsers. In *Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: Main Proceedings* (HLT/NAACL-03), Edmonton, Canada, pp. 236–243. Association for Computational Linguistics.

Steedman, Mark, Miles Osborne, Anoop Sarkar, Stephen Clark, Rebecca Hwa, Julia Hockenmaier, Paul Ruhlen, Steven Baker, and Jeremiah Crim (2003b). Bootstrapping statistical parsers from small datasets. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics* (EACL-03), Budapest, Hungary, pp. 331–338. Association for Computational Linguistics.

Strube, Michael (1998). Never look back: An alternative to centering. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics* (COLING/ACL-98), Montreal, Canada, pp. 1251–1257. Association for Computational Linguistics.

Strube, Michael and Christoph Müller (2003). A machine learning approach to pronoun resolution in spoken dialogue. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics* (ACL-03), Sapporo, Japan, pp. 168–175. Association for Computational Linguistics.

Strube, Michael, Stefan Rapp, and Christoph Müller (2002). The influence of minimum edit distance on reference resolution. In *Proceedings of the 2002 Conference*

on Empirical Methods in Natural Language Processing (EMNLP-02), Philadelphia, PA, pp. 312–319. Association for Computational Linguistics.

Strube, Michael and Maria Wolters (2000). A robust genre-independent model of pronominalization. In *Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics* (NAACL-00), Seattle, WA, pp. 18–25. Association for Computational Linguistics.

Taskar, Ben, Pieter Abbeel, and Daphne Koller (2002). Discriminative probabilistic models for relational data. In A. Darwiche and N. Friedman (Eds.), *Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence* (UAI-02), Edmonton, Canada, pp. 485–492. Morgan Kaufmann.

Tetreault, Joel (1999). Analysis of syntax-based pronoun resolution methods. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics* (ACL-99), College Park, MD, pp. 602–605. Association for Computational Linguistics.

Tetreault, Joel (2001). A corpus-based evaluation of centering and pronoun resolution. *Computational Linguistics 27*(4), 507–520.

Thompson, Cynthia A., Raymond J. Mooney, and Lappoon R. Tang (1997). Learning to parse natural language database queries into logical form. In *Proceedings of the ML-97 Workshop on Automata Induction, Grammatical Inference, and Language Acquisition*, Nashville, TN.

van Deemter, Kees and Rodger Kibble (2000). On coreferring: Coreference in MUC and related annotation schemes. *Computational Linguistics 26*(4), 629–637.

Vieira, Renata and Massimo Poesio (2000a). An empirically-based system for processing definite descriptions. *Computational Linguistics 26*(4), 539–593.

Vieira, Renata and Massimo Poesio (2000b). Processing definite descriptions in corpora. In S. Botley and A. McEnery (Eds.), *Corpus-based and Computational Approaches to Discourse Anaphora*, pp. 189–212. UCL Press.

Vilain, Marc, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman (1995). A model-theoretic coreference scoring scheme. In *Proceedings of the Sixth Message Understanding Conference* (MUC-6), Columbia, MD, pp. 45–52. Morgan Kaufmann.

Walker, Marilyn (1989). Evaluating discourse processing algorithms. In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics* (ACL-89), Vancouver, Canada, pp. 251–261. Association for Computational Linguistics.

Webber, Bonnie Lynn (1979). *A Formal Approach to Discourse Anaphora*. Garland Publishing, Inc.

Yang, Xiaofeng, Guodong Zhou, Jian Su, and Chew Lim Tan (2003). Coreference resolution using competitive learning approach. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics* (ACL-03), Sapporo, Japan, pp. 176–183. Association for Computational Linguistics.